QuestionIT

ReST Service Definition

In this document you will find a description of all the Endpoints as defined in backend of the QustionIT tool. For the POST and PUT commands a description is given of the JSON objects that have to sent. Before getting into detail about the Endpoints, first some information about the Enums that are used. This important because when you sent a JSON object that requires an Enum, you can choose to send a number or a specific string.

Enums

EnumExams

```
0  :  "NONE
1  :  "OCA"
2  :  "OCP"
3  :  "_70_480"
```

EnumLanguages

```
0  :  "NONE"
1  :  "JAVA"
2  :  "HTML_CSS_JS"
```

QuestionEndpoint

GET all Questions

Command: GET
Path: `'api/questions'`
Return: 200 *(ok)* + JSON([QuestionsModelBasic]) or 404 *(Not Found)*
Note:
- For the Creator only the id and name are returned

GET one Question

Command: GET
Path: `'api/questions/{id}'`
Return: 200 *(ok)* + JSON(QuestionModelBasic) or 404 *(Not Found)*
Note:
- For the Creator only the id and name are returned

GET one Question ModelExam

Command: GET
Path: `'api/questions/exam/{id}'`
Return: 200 *(ok)* + JSON(QuestionModelExam) or 404 *(Not Found)*

Notes:
- retrieves only fields applicable for making an exam. These are:

```
"id" : number,
"name" : "string",
"question" : "string",
"possibleAnswers" : ["string", "string"],
"typeOfQuestion" : "string",
"enabled" : boolean
```

GET one Question ModelExamReview
Command: GET
Path: `'api/questions/examreview/{id}'`
Return: 200 *(ok)* + JSON(QuestionModelExamReview) or 404 *(Not Found)*
Notes:
- retrieves only fields applicable for review an exam. These are:

```
"id" : number,
"name" : "string",
"question" : "string",
"possibleAnswers" : ["string", "string"],
"typeOfQuestion" : "string",
"enabled" : boolean,
"correctAnswers": { "id": number, "answers": [boolean, boolean]},
"explanationAnswer": "string"
```

Create a new Question
Command: POST
Path: `'api/questions/creator/{instructor_id}/correct-answers/{answerlist_id}'`
Return: 200 *(ok)* + id, 404 *(Not Found)* if answerlist or instructor do not exist, otherwise 500 *(Internal Server Error)*
Notes:
- Exclude 'creator', 'correctAnswers' and 'givenAnswers' from JSON
- If an existing 'id' is included than the question is made obsolete and a new one is created
- If non-existing 'id' is included it is ignored
- For the new question:
  - obsolete is set to false,
  - enabled is set to true and
  - creationDateTime is set to current time

```
{
  "name": "string",
  "programmingLanguage": EnumLanguages,
  "forExam": EnumExams,
  "question": "string",
  "possibleAnswers": [
    "string",
    "string"
  ],
  "explantionAnswer": "string",
  "typeOfQuestion": "string"
}
```

Update CorrectAnswers for existing Question
Command: POST

Path: `'api/question/{id}/correct-answers/{answerlist_id}'`
Return: 200 *(ok)* or 404 *(Not Found)*
Notes:
- No JSON object needs to be included
- If both the AnswerList and the Question exist, the AnswerList is attached

Add AnswerList to GivenAnswerList for existing Question
Command: POST
Path: `'api/questions/{id}/given-answers/{answerlist_id}'`
Return: 200 *(ok)* or 404 *(Not Found)*
Notes:
- No JSON object needs to be included
- If both the AnswerList and the Question exist, the AnswerList is attached

Delete an existing Question
Command: DELETE
Return: 200 *(ok)* or 404 *(Not Found)*
Path: `'api/questions/{id}'`

AnswerListEndpoint

GET one AnswerList
Command: GET
Path: `'api/questions/{id}'`
Return: 200 *(ok)* + JSON(AnswerList) or 404 *(Not Found)*
Note:
- For the Creator only the id and name are returned

Create a new AnswerList
Command: POST
Path: `'api/questions/answer-list'`
Return: 200 *(ok)* + id or 500 *(Internal Server Error)*
Notes:
- In the database, a list always consists of 10 answers. If the posted list containts less than 10 answers, the remainder are set to false
- If an existing 'id' is included than the AnswerList is updated
- If non-existing 'id' is included a new AnswerList is created

```
{
  "answers": [
    boolean,
    boolean,
    boolean,
    boolean,
    boolean,
    boolean,
    boolean,
    boolean,
    boolean,
    boolean
  ]
```

```
}
```

<<<< DOCUMENT BELOW NEEDS TO BE UPDATED >>>>

QuestionListEndpoint

GET all QuestionLists
Command: GET
Path: `'api/questionlists'`

GET one QuestionList
Command: GET
Path: `'api/questionlists/{id}'`

Create new or change one QuestionList
Command: POST
Path: `'api/questionlists'`
Notes:
- Exclude 'creator' and 'questions' from JSON
- If an existing 'id' is included than the data will be overwritten
- If non-existing 'id' is included it is ignored

```
{
  "name": "string",
  "programmingLanguage": EnumLanguages,
  "forExam": EnumExams,
  "creationDateTime": number,
  "examTimeInMinutes": number
}
```

Add new Question to existing QuestionList
Command: POST
Path: `'api/questionlists/{id}/question'`
Notes:
- Exclude 'creator', 'correctAnswers' and 'givenAnswers' from JSON
- If an existing 'id' is included than the data will be overwritten
- If non-existing 'id' is included, there is error ("500 Internal Server Error")

```
{
  "name": "string",
  "programmingLanguage": EnumLanguages,
  "forExam": EnumExams,
  "creationDateTime": number,
  "question": "string",
  "explantionAnswer": "string",
  "typeOfQuestion": "string"
}
```

Add existing Question to existing QuestionList
Command: POST
Path: `'api/questionlists/{id}/question/{question_id}'`

Note:
- No JSON object needs to be included
- If both the Question and the QuestionList exist, the Question is attached

Add existing Instructor to existing QuestionList
Command: POST
Path: `'api/questionlists/{id}/instructor/{instructor_id}'`
Note:
- No JSON object needs to be included
- If both the Instructor and the QuestionList exist, the Instructor is attached

Used cases:

1) Als een student op start drukt:
- post naar de backend: /exams/start/{questionlist_id}/{student_id}
- de backend maakt een nieuw examen aan op basis van de {questionlist_id}
- de backend voegt de id van het nieuwe examen toe aan "exams" van Student
- de backend bepaalt de tijd en slaat deze op in "startDateTime"
- de backend geeft de id van het examen terug naar de frontend
- de backend maakt voor elke vraag een AnswerList aan in het nieuwe Exam
- get naar de backend: /exams/{id}/question/1
- de backend stuurt een JSON Question terug van het Model Exam
- get naar de backend: /exams/{id}/answerlist/1
- de backend stuurt de JSON AnswerList
2) Elke 10 seconden stuurt de frontend een ping of iets dergelijks
- post naar de backend: /exams/{id}/ping
- de backend verhoogt "timeToCompleteInSeconds" sinds de laatste ping of start
3) Zodra een student een vraag markeert:
- post naar de backend: /exams/{id}/mark/{question_nr}
- backend voegt vraag nr toe aan markedQuestions list
4) Zodra een student op next klikt voor een volgende vraag:
- put JSON AnswerList naar de backend: /exams/{id}/answerlist/{answerlist_nr}
- get voor een nieuwe vraag: /exams/{id}/question/{question_nr}
- de backend stuurt een question terug van het Model Exam
- get naar de backend: /exams/{id}/answerlist/{answerlist_nr}
- de backend stuurt de JSON AnswerList
5) Zodra een student een vraag verlaat anders dan voor een volgende vraag:
- put JSON AnswerList naar de backend: /exams/{id}/answerlist/{answerlist_nr}
6) Als een student een gemarkeerde vraag opvraagt of naar een specifieke vraag wil gaan:
- get voor de vraag: /exams/{id}/question/{question_nr}
- de backend stuurt een question terug van het Model Exam
- get naar de backend: /exams/{id}/answerlist/{answerlist_nr}
- de backend stuurt de JSON AnswerList
7) Als student zijn vragen wil reviewen:
- get voor de beantwoorde vragen: /exams/{id}/review
- backend geeft een lijst met vragen waar de {question_nr} in staan waar de true count ongelijk is aan het aantal in de correctAnswerList
- get voor de marked questions: /exams/{id}/markedQuestions
- backend geeft een lijst met vragen waar de {question_nr} in staan die eerder gemarkeerd zijn
8) Als een student zijn examen afrond
- post dat voltooid: /exams/{id}/end

- backend vult "endDateTime" en "timeToCompleteInSeconds" in
- backend voegt {answerlist_id} aan "givenAnswers" van elke vraag

Notes:
{question_nr} verwijst naar de plaats van de vraag in de QuestionList. De QuestionList vertaalt dit naar een {question_id}. {question_nr} telt zoals mensen en is gelijk aan element index +1
{answerlist_nr} verwijst naar de plaats van de vraag in de givenAnswersList. De givenAnswersList vertaalt dit naar een {answerlist_id}. {answerlist_nr} telt zoals mensen en is gelijk aan element index +1
{question_id} is de echte id van een vraag