Template code review

Overview

This is the main file of the Flask application, called “The upper-level interface of the REST API”. Basically, in this file distinguished which methods could be used by Clients.

Goals

Create lightweight REST API using microframework Flask. The main advantages are flexibility and minimalistic set of features.

Usage Scenarios

Clients sends HTTPS requests on the REST API to fetch any data from the server*.

Clients sends HTTPS request on the RESP API to update information in the server*.

**Clients are trusted token holders*

Class Diagrams

All classes are abstract and have only database structured representation.

Code

Categories for review:

1. Design decisions

NameTTT: You should use NORMAL_PROGRAMMING_LANGUAGE for 'yyy’ instead of PYTHON.

Answer: Didn't get this comment.

2. API design

2.1 Reviewer1: Your api doesn’t work when you try to add book with long description. Because of the length of url overflow.

Answer: Bugs aren't part of this code.

2.2.1 Reviewer2: I do not think that it happens because of URL overflow (it can process string much longer than 100 symbols - limit for this backend), it seems like Backend itself has some troubles with processing of long strings.

Answer: Bugs aren't part of this code.

P.S: Fixed

2.3.1 “Search” request finds documents by authors only, but not by title, for example.

Answer: Bugs aren't part of this code.

P.S: Fixed

3. Architecture (including inheritance hierarchy, if any)

3.1.1 Reviewer3: Shouldn't the response codes be extracted into a separate module?
Answer: They are. Currently all possible response codes are in the codes.py module.
3.2.1 Reviewer3: “create_document” and “change_document_data” methods contain more than 10 code lines which are the same.
Answer: It’s true. However, creating a function that will extract values from request can make a structure more complicated.

4. Implementation techniques, in particular choice of data structures and Algorithms

Reviewer1: When professor books a document for a week, you still return it to user as a book he is able to take. But when this user want to take a document, there is an error. This book isn’t taken, but user can’t take it too.
Answer: Bugs aren't part of this code.
P.S: Fixed

5. Exceptions handling - Contracts

Reviewer1: No eiffel contracts found. Please write them.

6. Programming style, names

6.1.1 Reviewer3: What was the basis to choose CamelCase naming for URL patterns? How is it justified?
Answer: There is no any reasonable justification for this, except that the CamelCase is shorter than a Snake.
6.2.1 Reviewer3: Different naming style for semantically similar methods. In particular, “int_converter”, “boolean”, “int_converter_with_def_0” methods, which are all (supposedly) converters.
Answer: Yes, it’s true. I should check naming in conventers.

7. Comments and documentation

7.1.1 Reviewer2: You must properly comment your code for better understanding…
7.2 Reviewer1: With out coments it’s really hard to navigate inside your code
Answer: All documentation is in README file. You can find here a description of every method and parameter.