An-Najah National University
Department of Computer Engineering

Distributed Operating Systems

Homework 1

Abdallah Dereia 11718675 & Ali Bani Jaber 11743858

Jul 18 , 2021

Program Design:

Firstly to build our API we used Django Rest Framework because it is more suitable with SQL Databases and it is easy and powerful.

a. Then we started to work on the ***catalog server*** and implemented its functionality step by step as follows:

1. We created the needed endpoints which is represent the following functionality ***search(topic) , info(item_number) and decrement_number_of_books(item_number)*** as shown below :

```
catalog_server > urls.py > ...
1  from django.urls import path
2  from .views import search_book_by_topic, book_details, decrement_number_of_items
3
4  urlpatterns = [
5      path('search/<str:topic>', search_book_by_topic, name='search_by_topic'),
6      path('info/<int:pk>', book_details, name='book-details'),
7      path('update/<int:pk>', decrement_number_of_items, name='update_book'),
8  ]
```

2. we implemented the ***search(topic)*** functionality :

```python

@api_view(['GET'])
def search_book_by_topic(request, topic):

    '''
    search_book_by_topic Function:
        search for books with a specific topic

        Parameter:
            topic : book topic


        Return:
            returns array of books with the specified topic
    '''

    try:
        selected_books = Book.objects.filter(topic=topic)

        if not selected_books.exists():
            return Response({
                "Message": "There is no book available with the specified topic",
            }, status=status.HTTP_404_NOT_FOUND)

        serializer = BookSerializer(selected_books, many=True)

        return Response(serializer.data)

    except:
        return Response({
            "error": "An Error occured please try again later",
        }, status=status.HTTP_500_INTERNAL_SERVER_ERROR)

```

3. we implemented the ***info(item_number)*** functionality :

```python
@api_view(['GET'])
def book_details(request, pk):

    '''
    book_details Function:
        view specific book details

        Parameter:
            pk : book id

        Return:
            returns book details for the given id

    '''


    try:
        selected_book = Book.objects.get(pk=pk)

        serializer = BookSerializer(selected_book, many=False)
        return Response(serializer.data)

    except Book.DoesNotExist:
        return Response({
            "error": "There is no book available with the specified id",
        }, status=status.HTTP_404_NOT_FOUND)

    except:
        return Response({
            "error": "An Error occured please try again later",
        }, status=status.HTTP_500_INTERNAL_SERVER_ERROR)

```

4. we implemented the last endpoint which is ***decrement_number_of_books(item_number)***

```
92
93      try:
94          book = Book.objects.get(pk=pk)
95
96          book.number_of_items -= 1
97
98          book.save()
99
100         return Response({
101             "Message": "Book updated successfully",
102         })
103
104     except Book.DoesNotExist:
105         return Response({
106             "error": "There is no book available with the specified id",
107         }, status=status.HTTP_404_NOT_FOUND)
108
109     except:
110         return Response({
111             "error": "An Error occured please try again later",
112         }, status=status.HTTP_500_INTERNAL_SERVER_ERROR)
113
```

5. We deployed our catalog server on Heroku

b. Then we started to work on the ***Order server*** and added the only endpoint which is ***purchase(item_number):***

order_server > urls.py > ...

```
1  from django.urls import path
2  from .views import purchase_book
3
4  urlpatterns = [
5      path('purchase/<int:book_id>', purchase_book, name='purchase-book')
6  ]
```

then we implemented the functionality of the purchase(item_number):

order_server > views.py > ...

```python
from django.shortcuts import render
from rest_framework.response import Response
from rest_framework.decorators import api_view
from rest_framework import status
import requests
from .models import Order
import datetime

# Create your views here.


def query_book(book_id):
    try:
        response = requests.get(
            f'https://bazar-store-catalog.herokuapp.com/catalog/info/{book_id}')
        return response
    except:
        raise requests.ConnectionError


def book_exists(status_code):
    if status_code == 200:
        return True
    return False


def book_available_in_stock(number_of_items):
    print(number_of_items)
    if number_of_items > 0:
        return True
    return False


def decrement_number_of_books(book_id):
    requests.put(f'https://bazar-store-catalog.herokuapp.com/catalog/update/{book_id}')


```

```python
def store_order(book_id):
    order = Order(book_id=book_id)
    order.save()
    return


@api_view(['POST'])
def purchase_book(request, book_id):
    '''
    purchase_book Function:
        purchase a book with the specified id


        Parameter:
            book_id : book id

        Return:
            returns a successfull message alongside with the purchased book info
    '''

    try:
        response = query_book(book_id)

        if book_exists(response.status_code):
            book = response.json()
            if book_available_in_stock(book['number_of_items']):
                decrement_number_of_books(book_id)
                store_order(book_id)
                return Response({
                    "Message": "Book purchased successfully",
                    "book": book
                })

            return Response({
                "Message": "This Book is not available in the stock, sorry!"
            }, status=status.HTTP_404_NOT_FOUND)

```

Improvement Suggestion

I think that we can reduce the number of requests from order server to catalog server by just send one request and this request checks for book availability and at the same time update the number of books.

Endpoints Reference :

a. Catalog Server:

1. GET https://bazar-store-catalog.herokuapp.com/catalog/search/{topic name}
2. GET https://bazar-store-catalog.herokuapp.com/catalog/info/{item_number}
3. PUT https://bazar-store-catalog.herokuapp.com/catalog/update/{item_number}

b. Order Server:

1. POST https://bazar-order-server.herokuapp.com/order-server/purchase/{item_number}

```
74
75              return Response({
76                  "Message": "This Book is not found"
77              }, status=status.HTTP_404_NOT_FOUND)
78
79          except requests.ConnectionError:
80              return Response({
81                  "Message": "This service is not available right now"
82              }, status=status.HTTP_503_SERVICE_UNAVAILABLE)
83
```

```
38  def store_order(book_id):
39      order = Order(book_id=book_id)
40      order.save()
41      return
42
43
44  @api_view(['POST'])
45  def purchase_book(request, book_id):
46      '''
47      purchase_book Function:
48          purchase a book with the specified id
49
50
51          Parameter:
52              book_id : book id
53
54          Return:
55              returns a successfull message alongside with the purchased book info
56      '''
57
58      try:
59          response = query_book(book_id)
60
61          if book_exists(response.status_code):
62              book = response.json()
63              if book_available_in_stock(book['number_of_items']):
64                  decrement_number_of_books(book_id)
65                  store_order(book_id)
66                  return Response({
67                      "Message": "Book purchased successfully",
68                      "book": book
69                  })
70
71              return Response({
72                  "Message": "This Book is not available in the stock, sorry!"
73              }, status=status.HTTP_404_NOT_FOUND)
74
```

order_server > views.py > ...

```python
from django.shortcuts import render
from rest_framework.response import Response
from rest_framework.decorators import api_view
from rest_framework import status
import requests
from .models import Order
import datetime

# Create your views here.


def query_book(book_id):
    try:
        response = requests.get(
            f'https://bazar-store-catalog.herokuapp.com/catalog/info/{book_id}')
        return response
    except:
        raise requests.ConnectionError


def book_exists(status_code):
    if status_code == 200:
        return True
    return False


def book_available_in_stock(number_of_items):
    print(number_of_items)
    if number_of_items > 0:
        return True
    return False


def decrement_number_of_books(book_id):
    requests.put(f'https://bazar-store-catalog.herokuapp.com/catalog/update/{book_id}')


```