**Back-End Test Automation – Exam**



Submit your work as a **single zip / rar / 7z archive** holding your solutions for each problem at SoftUni Website.

Please refer to the end of this document for **instructions on how to submit your work**.

## The "Revue Crafters" System

**"Revue Crafters"** is an interactive web application for **sharing and managing revues**. It is accessible through a dedicated URL; the platform is designed for people to connect and share their experinces with movies, books, products, etc.. It offers a seamless experience with features like **revue** **creation** and **revue management**.

**Your task** is focused on **using Postman**, **Newman and RestSharp** to conduct **API tests**, ensuring the application's functionality works as expected.  
You can find the Web App here:

[**https://d3s5nxhwblsjbi.cloudfront.net/**](https://d3s5nxhwblsjbi.cloudfront.net/)

### API Endpoints

**"Revue Crafters"** exposes a **RESTful API**, available at**:**   
[**https://d2925tksfvgq8c.cloudfront.net**](https://d2925tksfvgq8c.cloudfront.net)

The **supported API endpoints** and **the interactive documentation** can be found at:

[**https://d2925tksfvgq8c.cloudfront.net/swagger/index.html**](https://d2925tksfvgq8c.cloudfront.net/swagger/index.html)

For your convenience, here is a **brief overview of the most important endpoints** below as well:

### 1. User

* POST /api/User/Create - create a new user. Post a JSON object in the request body:  
  {   
  "userName": "string",   
  "email": "user@example.com",   
  "password": "string",   
  "rePassword": "string",   
  "acceptedAgreement": true  
  }
* POST /api/User/Authentication - log in an existing user. Post a JSON object in the request body:  
  {  
  "email": "user@example.com",  
  "password": "string"  
  }

### 2. Access Token

* When a user logs in, the response format is JSON object:  
  **{**"email": " user@example.com",  
  **"password": "string",  
  "accessToken": "eyJhbGciOiJ…"  
  }**

**NB! Access token is needed for all revue requests. It should be placed under the Authorization tab, Bearer Token option.**

### 3. Revue

All of the **following requests require Authorization**!

* **GET /api/Revue/All** – list all revues (empty request body).
* **POST** **/api/Revue/Create** – create a new revue.  
  Include a JSON object in the request body (title and description are mandatory, url is optional):   
  **{  
  "title": "string",  
  "url": "",  
  "description": "string"  
  }**
* PUT /api/Revue/Edit – replace the existing revue with the new one.  
  Include a JSON object in the request body (title and description are mandatory, url is optional):   
  **{  
  "title": "string",   
  "url": " ",   
  "description": "string"  
  };**

Requires **queryParameter: ?revueId={revueId}**

* DELETE **/api/Revue/Delete** – delete existing revue;   
  **Requires queryParameter: ?revueId={revueId}**

## RESTful API: Postman API Tests (35 points)

Your task is to write **API tests** with Postman for certain **RESTful API endpoints**. Organize your tests within a collection, **use collection variables** and **pre-request scripts** to **guarantee successful execution on every run**. **It's important to use collection variables**, **NOT ENVIRONMENT VARIABLES**, to maintain the integrity and portability of the test suite.

### Prerequisites

First you need to **register a new user**. **Registration** of a **new user** is **a mandatory step** that you must complete prior to conducting your API tests. You have the **flexibility to register** either through the [**web UI**](https://d3s5nxhwblsjbi.cloudfront.net/) or **by making a request via Postman**. Please note that this **initial registration process is not included in the scope of your assignment and will not contribute to your final score**. However, it is essential as you will **need an active user account** for all subsequent API requests that form the core of your test cases.   
**If you decide to register via Postman**, **remove this request from your collection.**

### Base Setup

* Add the base URL [**https://d2925tksfvgq8c.cloudfront.net/**](https://d2925tksfvgq8c.cloudfront.net/) as a collection variable **{baseURL}**.
* Ensure all requests use this **{baseURL}**.

### Login and Authentication

* Send a **POST request** for **user authentication**.
* **Assert a 200 status** code for success.
* **Assert** that the **response body includes** the attributes **email**, **password**, and **accessToken**. The objective is not to confirm the specific content of these fields but to ensure that they are present in the response.
* Save the **accessToken** as a **collection variable** **{{token}}** for **Bearer Token authorization in subsequent requests**.

### Create a New Revue

* Use a **pre-request script** to **generate a random title for the revue** (**a word followed by up to three digits**).
* Store this title as a **{{randomTitle}}** collection variable.
* **Send a POST request** with **{{randomTitle}}** and a **description** (description can be added manually).
* **Assert a 200 status** code.
* **Assert the "Successfully created!" message** in the response body.
* **Assert** that the **title** of the **created revue** in the response matches the expected title.

### List all Revues

* **Send a GET request to receive a list of all created revues**.
* **Assert a 200 status** code.
* **Assert** that the **response is an array** and that it contains **at least one item**.
* Extract the **id of the last created revue** from the response body and **store it as a collection variable** **{{lastRevueId}}.**

### Edit the Created Revue

* **Send a PUT request** to **modify the revue** identified by **{{lastRevueId}}.** **Change its title** (you can do this manually, no need for scripting).
* **Assert a 200 status** code.
* **Assert** the **"Edited successfully"** message.
* **Assert** that the **title in the response matches the new title you provided.**

### Delete the Edited Revue

* Send a **DELETE request to delete the edited Revue** identified by **{{lastRevueId}}.**
* **Assert a 200 status** code.
* **Assert** the **"The revue is deleted!"** message.

### Final Steps

1. Make sure that your collection contains all the requests needed:

* **Login**
* **Create New Revue**
* **List All Revues**
* **Edit the Created Revue**
* **Delete the Created Revue**

1. Make sure that the collection can be executed successfully on each run.
2. Export and save your collection in a single file.

## Newman with htmlextra Reporter (15 points)

* **Run** the exported **collection** that you created via Postman in **Newman**.
* Use **htmlextra as a reporter.**
* Add the **generated html report** to the archive with your other tasks.

## RESTful API: RestSharp API Tests (50 points)

**In this task**, you will demonstrate your ability to interact with a **RESTful API** using **RestSharp** within a **.NET test project**. Your primary goal is to create a set of **automated tests from scratch** that validate the key functionalities of the **RevueCrafters API**. You will be **assessed** on your ability to configure a **test project**, **utilize RestSharp** to **make API requests**, and **assert** the expected **responses using NUnit**.

### 3.0. Prerequisites

First, you are required to **set up a new NUnit Test Project** in your Visual Studio. Ensure you **install all necessary packages**, **including RestSharp**, to create a functional API testing suite. This project will serve as the foundation for your subsequent testing tasks.

### 3.1. Base Setup

* **Initialize a RestClient** with the **base URL of the API**.
* Since you've already have an account**, authenticate** with **your credentials**, and **store** the received **JWT token**.
* **Configure** the **RestClient with an Authenticator using the stored JWT token**.

### 3.2. Data Transfer Objects (DTOs)

**Before you begin writing your tests**, it's important to **create Data Transfer Objects (DTOs).** Given that you are **familiar** with the **structure of both the requests and responses**, you have the flexibility to **create as many DTOs as you need**. However, these **two DTOs should be sufficient** for the scope of your task:

* **ApiResponseDTO** - this DTO will be used to parse common response structures from the API. It should include the following properties:
* **Msg** of **type string** to capture response messages.
* **RevueId** of **type string** to capture the unique identifier of a revue. This field may be null for responses that do not include revue id.
* **RevueDTO** - representing the structure of a revue for creation and editing purposes. It should include the following properties:
  + **Title** of **type string** for the revue's title.
  + An **optional Url** of **type string** representing a link to the revue's picture, if applicable.
  + **Description** of **type string** for the revue's description.

### 3.3. Create a New Revue with the Required Fields

* **Create a test** to send a **POST request** to **add a new revue**.
* **Assert** that the response **status code is OK (200)**
* **Assert** that the **response message** indicates the revue was **"Successfully created!"**

### 3.4. Get All Revues

* **Create a test to send a GET request to list all revues.**
* **Assert that** the response **status code is OK (200)**
* **Assert that** the response contains a **non-empty array.**
* **Store** the **id** of the **last created revue** in a **static member of the test class to maintain its value between test runs**.

### 3.5. Edit the Last Revue that you Created

* Create a test that **sends a PUT request** to edit the revue.
* Use the **id** that you **stored in the previous request, as a query parameter**.
* **Assert** that the **response status code is OK (200)**
* **Assert** that the **response message** indicates the revue was **"Edited successfully"**

### 3.6. Delete the Revue that you Edited

* Create a test that **sends a DELETE request**.
* Use the **id** that you **stored in the "Get All Revues" request as a query parameter.**
* **Assert that** the response **status code is OK (200)**
* **Confirm** that the response message is **"The revue is deleted!"**

### 3.7. Try to Create a Revue without the Required Fields

* Write a test that attempts to **create a revue with missing required fields** (Title, Description).
* Send the **POST reques**t with **the incomplete data**.
* **Assert** that the response status code is **BadRequest (400)**

### 3.8. Edit a Non-existing Revue

* Write a test to **send a PUT request to edit a revue with a RevueId that does not exist**.
* **Assert** that the response status code is **BadRequest (400).**
* **Assert** that the response message indicates **"There is no such revue!"**

### 3.9. Delete a Non-existing Revue

* Write a test to **send a DELETE request to edit a revue with a RevueId that does not exist**.
* **Assert** that the response status code is **BadRequest (400)**
* **Assert** that the response message indicates **"There is no such revue!"**

### 3.10. Final Steps

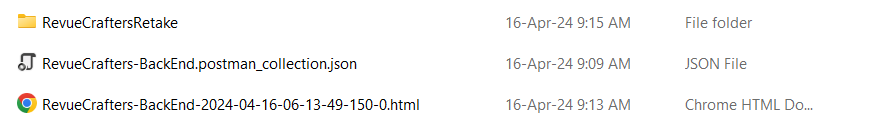
* Ensure that each test is correctly **ordered to maintain the required sequence of actions. Use [Order( )]**
* Verify that tests are designed to **run successfully in on each run.**
* **Delete bin and obj folders** from your solution folder.

## How to submit your exam You should attach a single zip / rar / 7z archive containing all of your tasks.

Upload your archive at SoftUni website, into [Retake Exam](https://softuni.bg/trainings/4399/back-end-test-automation-february-2024#lesson-65571) section.

* The Postman collection should be exported in a single **JSON** file.
* You also need to export the **html file** obtained **from the htmlextra reporter in Newman.**
* Your **RestSharp API Test** project should be **in a folder**.

At the end, the content of your archive should look similar:



Before archiving, please make sure that you **deleted all bin and obj folders from your RestSharp Test project.**