Nghề Thiết kế và phát triển trang Web

Module C - NodeJS

Thời gian: 2.0 giờ

VNSC2018_WebDesign_Module_C

INTRODUCTION

New startup company would like to create flight and hotel booking system website. This website should help user to book the flight and hotel. They ask you to create a website with separate architecture, server and client side. They're planning to create a mobile version later, so they're asking you to build this website with the web service architecture.
Development phase will be separate into two phases. First phase is creating backend web service and the second phase is to create front end website. The server should serve API only. There should be no any visual interface output. The only interface is the Application Programming Interface (API).

DESCRIPTION OF PROJECT AND TASKS

The description for this module will be listed below. Your first task is to create a restful web service API that could be used by the front end to communicate the data. The service is created with NodeJS.

The server takes application/json request. Response should also be in JSON format. All response should follows with a correct status code—Success, Not Found and Bad Request.
Data Type

- All "date" should return in String
- All "ID" should return in integer
- All fields without specifying the type is string.
- All "date" in requested parameters is formated as 'yyyy-mm-dd'

The followings are the list of web service that we need:

1. **Authentication**
   a. Login (*v1/auth/login*)
   Description: For client to get login token via username and password
   Request method: **POST**
   Header: header authorization basic
   Requested parameter:
   - username
   - password

   Response result:
   - If success,
     - header: response status: 200
     - body:
       - token`: authorization token (to be valid until logout). Token will be generated by the system from logged in username with md5 encryption method
       - Role (ADMIN / USER)
   - If username/password not correct or empty,
     - header: response status: 401
     - body: message: invalid login

   b. Logout (*v1/auth/logout?token={AUTHORIZATION_TOKEN}*)
   Description: For server to invalid the user's token
   Request method: **GET**
   Header: header authorization basic
   Response result:
   - If success,
     - header: response status: 200

- body
  - message: logout success
- If unauthorized user access it, data:
  - Message: Unauthorized user
  - Response status: 401

2. Airline

The API allows airlines company creation, flight creation, update, delete, flights query and booking of flights.

a. Create Airline company (*v1/airline?token={AUTHORIZATION_TOKEN}*), only admin access this API.

Description: For client to create a new airline company object.

Request method: **POST**

Header: header authorization basic

Requested parameter:

- airline_name
- city_name (which the airline is based at)

Response result:

- If success, body:
  - Message: create success
  - id: id of created Airline company
  - Response status: 200
- If failed, body: (name is duplicated)
  - Message: Data cannot be processed
  - Response status: 422
- If unauthorized user access it, body:
  - Message: Unauthorized user
  - Response status: 401

b. Create airline flight (*v1/flight?token={AUTHORIZATION_TOKEN}*), only admin access this API.

Description: For client to create a new airline flight object.

Request method: **POST**

Header: header authorization basic

Requested parameter:

- from_date
- to_date
- flight_time
- arrival_time
- from_city_name (departure city name),
- to_city_name (destination city name)
- airline_id
- price (fare price (interger))

Response result:

- If success, body:
  - Message: create success
  - id: id of created flight
  - Response status: 200
- If failed, body: (Wrong foreign key, wrong date format, wrong integer)
  - Message: Data cannot be processed
  - Response status: 422
- If unauthorized user access it, body:
  - Message: Unauthorized user
  - Response status: 401

c. Query/Get airline flights list (*v1/flight/{DEPARTURE_DATE}/{DEPARTURE_CITY_NAME}/{DESTINATION_CITY_NAME}?token={AUTHORIZATION_TOKEN}*).
Description: For client to list of airline flight are available on departure date, departure city name and destination city name.
Request method: **GET**
Header: header authorization basic
Response result:
- If success, array of flights. Each flight contains:
  - flight_id
  - flight_code
  - departure_time
  - arrival_time
  - airline_id
  - airline_name
  - all fields from requested parameters

  Response status: 200
- If failed, body: (wrong departure_date format)
  - Message: Data cannot be processed
  - Response status: 422
- If unauthorized user access it, body:
  - Message: Unauthorized user
  - Response status: 401

d. Book a flight (*v1/flight-book?token={AUTHORIZATION_TOKEN}*)
Description: For client to book a flight.
Request method: **POST**
Header: header authorization basic
Requested parameter:
  - flight_type (One way only or with return flight)
  - from_date
  - from_time
  - return_date (if not one way flight)
  - return_time (if not one way flight)
  - from_city_name
  - to_city_name
  - flight_class (Economy, Business or First class)
  - total_adults
  - total_children
  - passengers consist of first_name, last_name (array of passengers's first name and last name)

Response result:
- If success, body:
  - flight_book_id
  - All fields from Requested parameters
  - Response status: 200
- If failed, body: (wrong integer, date format)
  - Message: Data cannot be processed
  - Response status: 422
- If unauthorized user access it, body:
  - Message: Unauthorized user
  - Response status: 401

e. Update airline flight (*v1/flight/{ID}?token={AUTHORIZATION_TOKEN}*), only admin access this API.
Description: For client to update an existing airline flight object via flight ID.
Request method: **PUT**
Header: header authorization basic
Requested parameter:
  - from_date
  - to_date

- flight_time
- arrival_time
- from_city_name (departure city name),
- to_city_name (destination city name)
- airline_id
- price (fare price (interger))

Response result:

- If success, body:
  - Message: update success
  - Response status: 200
- If failed, body: (wrong integer, wrong date format, wrong foreign key, wrong id)
  - Message: Data cannot be updated
  - Response status: 400
- If unauthorized user access it, body:
  - Message: Unauthorized user
  - Response status: 401

f. Delete airline flight (*v1/flight/{ID}?token={AUTHORIZATION_TOKEN}*), only admin access this API.

Description: A request to delete a airline flight object via flight ID.

Request method: **DELETE**

Header: header authorization basic

Response result:

- If success, body:
  - Message: delete success
  - Response status: 200
- If failed, body: (wrong id)
  - Message: Data cannot be deleted
  - Response status: 400
- If unauthorized user access it, body:
  - Message: Unauthorized user
  - Response status: 401

3. Hotel

The API allows hotel creation, update, delete, hotel query and booking of hotel.

a. Create hotel (*v1/hotel?token={AUTHORIZATION_TOKEN}*), only admin access this API.

Description: For client to create a new hotel object.

Request method: **POST**

Header: header authorization basic

Requested parameter:

- name
- city_name
- description
- room_types consist name, price (array of room types, each type includes a name and a price),
- images (array of image URLs)
- capacity

Response result:

- If success, body:
  - Message: create success
  - Response status: 200
- If failed, body: (name is duplicated, wrong integer)
  - Message: Data cannot be processed
  - Response status: 422
- If unauthorized user access it, body:
  - Message: Unauthorized user
  - Response status: 401

b. Query/Get hotel list (*v1/hotel/{CITY_NAME}?token={AUTHORIZATION_TOKEN}*)

Description: For client to list of hotel are available on city name.

Request method: **GET**

Header: header authorization basic

Response result:

- If success, array of hotels. Each hotel contains:
  - hotel_id
  - hotel_name
  - city_name
  - description
  - room_types consist name, price (array of room types, each type includes a name and a price),
  - images (array of image URLs)
  - room_capacity

  Response status: 200
- If failed, body:
  - Message: Data cannot be processed
  - Response status: 422
- If unauthorized user access it, body:
  - Message: Unauthorized user
  - Response status: 401

c. Book a hotel (*v1/hotel-book?token={AUTHORIZATION_TOKEN}*)

Description: For client to book a hotel.

Request method: **POST**

Header: header authorization basic

Requested parameter:

- check_in_date
- check_out_date
- hotel_id
- total_guests
- guests consist first_name, last_name (array of guest's first name and last name)
- total_rooms
- room_type (name of room type)

Response result:

- If success, body:
  - hotel_book_id
  - All fields from requested parameters
  - Response status: 200
- If failed, body: (wrong integer, wrong date format, wrong foreign key)
  - Message: Data cannot be processed
  - Response status: 422
- If unauthorized user access it, body:
  - Message: Unauthorized user
  - Response status: 401

d. Update hotel (*v1/hotel/{ID}?token={AUTHORIZATION_TOKEN}*), only admin access this API.

Description: For client to update an existing hotel object via hotel ID.

Request method: **PUT**

Header: header authorization basic

Requested parameter:

- name
- city_name
- description
- room_types consist name, price (array of room types, each type includes a name and a price),
- images (array of image URLs)
- capacity

Response result:

- If success, body:
  - Message: update success

- Response status: 200

- If failed, body: (wrong integer, wrong id)
  - Message: Data cannot be updated
  - Response status: 400
- If unauthorized user access it, body:
  - Message: Unauthorized user
  - Response status: 401

e. Delete hotel (*v1/hotel/{ID}?token={AUTHORIZATION_TOKEN}*), only admin access this API.
Description: A request to delete a hotel object via hotel ID.
Request method: **DELETE**
Header: header authorization basic
Response result:

- If success, body:
  - Message: delete success
  - Response status: 200
- If failed, body: (wrong id)
  - Message: Data cannot be deleted
  - Response status: 400
- If unauthorized user access it, body:
  - Message: Unauthorized user
  - Response status: 401

4. Transaction

The API allows transaction creation.

a. Create a transaction (*v1/transaction?token={AUTHORIZATION_TOKEN})*
Description: For client to create a new transaction.
Request method: **POST**
Header: header authorization basic
Requested parameter:

- title (Mr, Mrs, Miss, Dr)
- first_name (User's first name)
- last_name (User's last name)
- email
- phone
- payment_method (transfer, credit card or paypal)
- card_name (if pay by credit card)
- card_number (if pay by credit card)
- ccv (if pay by credit card)

Response result:

- If success, body:
  - payment_status (payment status to indicate success or error)
  - transaction_id
  - Response status: 200
- If failed, body: (wrong email, card_number, ccv)
  - Message: Data cannot be processed
  - Response status: 422
- If unauthorized user access it, body:
  - Message: Unauthorized user
  - Response status: 401

Notes

- Competitors should implement a minimum of one of the server-side frameworks/libraries that are provided.
- The specified database tables need to be implemented. More tables may be added if needed. Provide a final SQL-dump and ERD screen as specified below.
  All API should fulfill all requirements as stated in the description. All prefix, RESTful-URL and HTTP-Method from given API link should be implemented correctly and not be changed. If needed, you may add other API, besides all API that already mentioned in this document.

- Create the following users to login to the system:
  - Admin with username: `admin` and password: `adminpass`,
  - User1 with username: `user1` and password: `user1pass`,
  - User2 with username: `user2` and password: `user2pass`

INSTRUCTIONS TO THE COMPETITOR

- Competitor is provided with NodeJS framwork **node-v8.9.3.zip**
- Save the files of your application in directory on the server called "**XX_Server_C**", where XX is your competitor ID.
- You should export the latest SQL schema as a MySQL dump file.
  - ERD screen shot named "**XX_ERD.png**" in "db-dump" folder inside of **XX_Server_C**
  - Database dump named "**XX_database.sql**" in "db-dump" folder inside of **XX_Server_C**