SOAL
Module SERVER

DESKRIPSI TEKNIS

WEB TECHNOLOGIES

MODULE SERVER

CONTENTS

This module has the following files:

- MODULE_SERVER.doc
- MODULE_SERVER_MEDIA.zip

INTRODUCTION

Trilu, task management website, ask you to make their minimum viable product. Your task is to implement the backend with Laravel PHP Framework and frontend with JavaScript Framework (VueJS, AngularJS, Angular, or ReactJS). The front-end design skeleton is provided. The detail description and tools that you can use will be described below.

DESCRIPTION OF PROJECT AND TASKS

API List:

Use provided ERD to make your database. Create dummy users on users table (password is hashed using bcrypt):

username	password	First name	Last name
john.doe	12345	John	Doe
richard.roe	12345	Richard	Roe
jane.poe	12345	Jane	Poe

These are the list of web service endpoint that requested by **Trilu**:

1. **Authentication**
    a. **Register**

    URL: [domain]/v1/auth/register

    Description: For client to register new user

    Method: POST

    Request Parameter:

    - Body:
        - First Name
        - Last Name
        - Username
        - Password

    Validation:

    - First Name must be alphabet only, length between 2 – 20
    - Last name must be alphabet only, length between 2 – 20
    - Username only consist of alphanumeric, underscore '_', or dot '.', length between 5 – 12
    - Username must be unique
    - Password length between 5 - 12

    Response:

    - If register success, registered user automatically logged in:

Header: response status: 200

Body:

  - Token (authorization token generated by the system from logged in user id with bcrypt hashing method)
  - Role

- If input validation failed:

  Header: response status: 422

  Body: message: invalid field

b. Login

URL: [domain]/v1/auth/login

Description: For client to generate and get login token using username and password. Username and password must be valid.

Method: POST

Request Parameter:

- Body:
  - Username
  - Password

Response:

- If login success:

  Header: response status: 200

  Body:

  - Token (authorization token generated by the system from logged in user id with bcrypt hashing method)
  - Role

- If login failed (username or password do not match or empty):

  Header: response status: 401

  Body: message: invalid login

c. Logout

URL: [domain]/v1/auth/logout?token={*AUTHORIZATION_TOKEN*}

Description: For server to make token invalid

Method: GET

Response:

- If logout success:

  Header: response status: 200

  Body:

  - message: logout success

- If logout failed (token invalid):

  Header: response status: 401

  Body: message: unauthorized user

2. **Board**

   a. **Create new board**

   URL: [domain]/v1/board?token={*AUTHORIZATION_TOKEN*}

   Description: For client to create new board. Assign creator as team member when board created.

   Method: POST

   Request Parameter:

   - Body:
     - Name

   Validation:

   - name must be filled

   Response:

   - If success:
     Header: response status: 200
     Body: message: create board success
   - If input validation failed:
     Header: response status: 422
     Body: message: invalid field
   - If unauthorized user access it (only logged in user can access this endpoint):
     Header: response status: 401
     Body: message: unauthorized user

   b. **Update board**

   URL: [domain]/v1/board/{*board_id*}?token={*AUTHORIZATION_TOKEN*}

   Description: For client to update existing board by id

   Method: PUT

   Request Parameter:

   - Body:
     - Name

   Validation:

   - name must be filled

   Response:

   - If success:
     Header: response status: 200
     Body: message: update board success
   - If input validation failed:
     Header: response status: 422
     Body: message: invalid field
   - If unauthorized user access it (only team member can access this endpoint):
     Header: response status: 401
     Body: message: unauthorized user

c. **Delete board**

URL: [domain]/v1/board/{*board_id*}?token={*AUTHORIZATION_TOKEN*}

Description: For client to delete existing board by id

Method: DELETE

Response:

- If success:

  Header: response status: 200

  Body: message: delete board success
- If unauthorized user access it (only creator can access this endpoint):

  Header: response status: 401

  Body: message: unauthorized user

d. **Get all boards**

URL: [domain]/v1/board?token={*AUTHORIZATION_TOKEN*}

Description: For client to get all boards data based on logged in user, as a member or creator.

Method: GET

Response:

- If success:

  Header: response status: 200

  Body: all board data in json (consists of id, name, creator_id)
- If unauthorized user access it (only logged in user can access this endpoint):

  Header: response status: 401

e. **Open board**

URL: [domain]/v1/board/{*board_id*}?token={*AUTHORIZATION_TOKEN*}

Description: For client to get board detail based on board_id. Lists and cards sorted by column order.

Method: GET

Response:

- If success:

  Header: response status: 200

  Body: eager load all board data in json (consists of all team members, all lists of board and cards inside list)

Format:

```
{
  “id”: [board_id],
  “name”: [board_name],
   creator_id: [creator_id],
   “members”: [
      {
         “id”: [user_id],
         “first_name”: [first name],
         “last_name”: [last name],
         “initial”: [generated from first name and last name]
      },
      …
   ],
   “lists”: [
      {
         “id”: [list_id],
         “name”: [list_name],
         “order”: [order],
         “cards”: [
            {
               “card_id”: [card_id],
               “task”: [card_task],
               “order”, [order]
            },
            …
         ]
      },
      …
   ]
}
```

- If unauthorized user access it (only team member can access this endpoint):
  Header: response status: 401

f. Add team member

URL: [domain]/v1/board/*{board_id}*/member?token={*AUTHORIZATION_TOKEN*}

Description: For client to add team member to the board.

Method: POST

Request Parameter:

- Body:
  - Username

Validation:

- username must be existing in user table

Response:

- If success:
  Header: response status: 200
  Body: message: add member success
- If input validation failed:
  Header: response status: 422
  Body: message: user did not exist
- If unauthorized user access it (only team member can access this endpoint):
  Header: response status: 401
  Body: message: unauthorized user

g. **Remove team member**

URL: [domain]/v1/board/*{board_id}*/member/*{user_id}*?token={*AUTHORIZATION_TOKEN*}

Description: For client to delete team member from the board using user id.

Method: DELETE

Response:

- If success:
  Header: response status: 200
  Body: me ssage: remove member success
- If unauthorized user access it (only team member can access this endpoint):
  Header: response status: 401
  Body: message: unauthorized user

3. **List**

a. **Create new list**

URL: [domain]/v1/board/*{board_id}*/list?token={*AUTHORIZATION_TOKEN*}

Description: For client to create new list to the board

Method: POST

Request Parameter:

- Body:
  - Name

Validation:

- name must be filled

Response:

- If success:
  Header: response status: 200
  Body: message: create list success

- If input validation failed:
  Header: response status: 422
  Body: message: invalid field
- If unauthorized user access it (only board team member can access this endpoint):
  Header: response status: 401
  Body: message: unauthorized user

b. Update list

URL: [domain]/v1/board/{*board_id*}/list/*{list_id}*?token={*AUTHORIZATION_TOKEN*}

Description: For client to update existing list by id

Method: PUT

Request Parameter:

- Body:
  - Name

Validation:

- name must be filled

Response:

- If success:
  Header: response status: 200
  Body: message: update list success
- If input validation failed:
  Header: response status: 422
  Body: message: invalid field
- If unauthorized user access it (only board team member can access this endpoint):
  Header: response status: 401
  Body: message: unauthorized user

c. Delete list

URL: [domain]/v1/board/{*board_id*}/list/*{list_id}*?token={*AUTHORIZATION_TOKEN*}

Description: For client to delete existing list by id

Method: DELETE

Response:

- If success:
  Header: response status: 200
  Body: message: delete list success
- If unauthorized user access it (only board team member can access this endpoint):
  Header: response status: 401
  Body: message: unauthorized user

d. **Move list to right**

URL: [domain]/v1/board/{*board_id*}/list/*{list_id}*/right?token={*AUTHORIZATION_TOKEN*}

Description: For client to switch the order of the selected list with the list on the right

Method: POST

Response:

- If success:
  Header: response status: 200
  Body: message: move success
- If unauthorized user access it (only board team member can access this endpoint):
  Header: response status: 401
  Body: message: unauthorized user

e. **Move list to left**

URL: [domain]/v1/board/{*board_id*}/list/*{list_id}*/left?token={*AUTHORIZATION_TOKEN*}

Description: For client to switch the order of the selected list with the list on the left

Method: POST

Response:

- If success:
  Header: response status: 200
  Body: message: move success
- If unauthorized user access it (only board team member can access this endpoint):
  Header: response status: 401
  Body: message: unauthorized user

4. **Card**

a. **Create new card**

URL: [domain]/v1/board/*{board_id}*/list/*{list_id}*/card?token={*AUTHORIZATION_TOKEN*}

Description: For client to create new card to the list

Method: POST

Request Parameter:

- Body:
  - Task

Validation:

- Task must be filled

Response:

- If success:
  Header: response status: 200
  Body: message: create card success

- If input validation failed:
  Header: response status: 422
  Body: message: invalid field
- If unauthorized user access it (only board team member can access this endpoint):
  Header: response status: 401
  Body: message: unauthorized user

b. Update card

URL:
[domain]/v1/board/{*board_id*}/list/*{list_id}*/card/*{card_id}*?token={*AUTHORIZATION_TOKEN*}

Description: For client to update existing card by id

Method: PUT

Request Parameter:

- Body:
  - Task

Validation:

- Task must be filled

Response:

- If success:
  Header: response status: 200
  Body: message: update card success
- If input validation failed:
  Header: response status: 422
  Body: message: invalid field
- If unauthorized user access it (only board team member can access this endpoint):
  Header: response status: 401
  Body: message: unauthorized user

c. Delete card

URL:
[domain]/v1/board/{*board_id*}/list/*{list_id}*/card/*{card_id}*?token={*AUTHORIZATION_TOKEN*}

Description: For client to delete existing card by id

Method: DELETE

Response:

- If success:
  Header: response status: 200
  Body: message: delete card success
- If unauthorized user access it (only board team member can access this endpoint):
  Header: response status: 401
  Body: message: unauthorized user

d. **Move up card**

URL: [domain]/v1/card/*{card_id}*/up?token={*AUTHORIZATION_TOKEN*}

Description: For client to switch the order of the selected card with the card above

Method: POST

Response:

- If success:
  Header: response status: 200
  Body: message: move success
- If unauthorized user access it (only board team member can access this endpoint):
  Header: response status: 401
  Body: message: unauthorized user

e. **Move down card**

URL: [domain]/v1/card/*{card_id}*/down?token={*AUTHORIZATION_TOKEN*}

Description: For client to switch the order of the selected card with the card below

Method: POST

Response:

- If success:
  Header: response status: 200
  Body: message: move success
- If unauthorized user access it (only board team member can access this endpoint):
  Header: response status: 401
  Body: message: unauthorized user

f. **Move card to another list**

URL: [domain]/v1/card/*{card_id}*/move/*{list_id}*?token={*AUTHORIZATION_TOKEN*}

Description: For client to move card to another list. List must be in the same board.

Method: POST

Response:

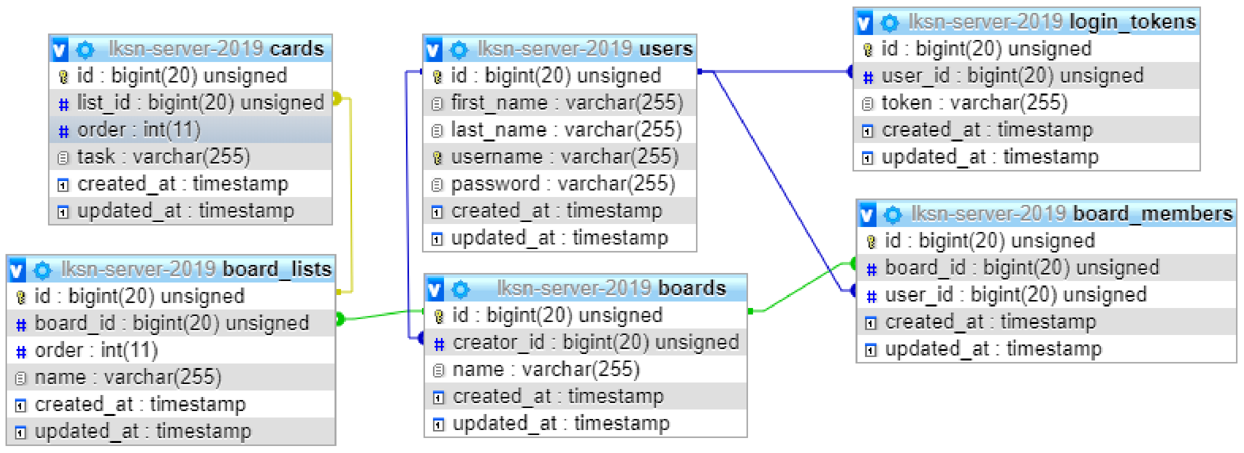
- If success:
  Header: response status: 200
  Body: message: move success
- If list id is not in the same board:
  Header: response status: 422
  Body: message: move list invalid
- If unauthorized user access it (only board team member can access this endpoint):
  Header: response status: 401
  Body: message: unauthorized user

Make sure that all validations are done in **both backend and frontend**. The complete minimum viable product for **Trilu** system should cover the following requirement:

Menu	**Detail**
Login & Register	• User can register into the system on the register page • User can login (and logout) into the system on the start page of the application • After login, user directed to “home” menu • Links to the Home and Logout are always visible on the top part of the page while the user is logged in
Home	• On “home” menu, user can add new board or access board if the user is the team member • User can update board’s name by clicking the board name, type new name, then press Enter to submit • Creator of the board can delete a board by deleting its entire name, then press Enter to submit • Make sure your system is preventing users to view and access unauthorized board
Board	• On “board” menu, user can manage board: members, lists, and cards **Member** • User can add new member using username • User can remove member by clicking its initial then click confirmation button **List** • User can add new list to the board • User can update list’s name by clicking the list name, type new name, then press Enter to submit • User can delete a list by deleting its entire name, then press Enter to submit • User can move list left and right **Card** • User can add new card to a list • User can update card’s task by clicking the card, type new task, then press Enter to submit • User can delete a card by deleting its entire task, then press Enter to submit • User can move card up and down in the list • User can move card to another list by click the card, then click another list.

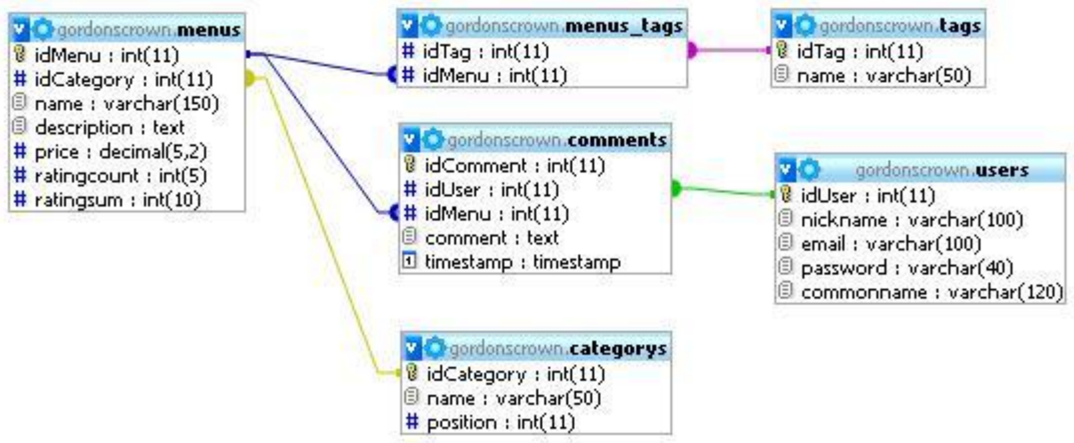
ERD

You can use and improve ERD below:



INSTRUCTIONS TO THE COMPETITOR

- Save your files in your root directory on the server called "**XX_SERVER_MODULE**" where XX is your computer number.
- Create/generate a DB-diagram named “db-diagram.xxx” (xxx is the extension/type of the file eg. pdf or jpg) and put it into the directory mentioned above. Example:



- For this module, you must use one of the four available frameworks provided. Applications developed without use of any of these frameworks will not be considered. You should take advantage of the framework as much as possible.
- Assessment will be done on the files and data in your database “**XX_SERVER**" on the central server.