Online Friendly
Skills Game 2021

TEST PROJECT
for Web design and development

WSA2021_TP17_Server_side_actual EN

Submitted by: Digital Skills Game Manager
Name: Konstantin Larin
Member Country or Region: Russia

CONTENTS

INTRODUCTION

Technologies of this module: REST API
Time to complete: 3 hours

You need to use all available skills in server development to create a REST API that will be responsible for controlling access to checkpoints at the customer's site.
The customer wants the API to be easy to maintain, so using frameworks would be a plus.

DESCRIPTION OF PROJECT AND TASKS

Your task is to implement a REST API that will meet the customer's requirements.

For your convenience, all URLs will use the {host} pseudonym, which indicates the address http://xxxxxx-m1.wsr.ru/, where xxxxxx is your participant login.

The customer wants to organize access control on their territory using control points. To do this, he needs an API with the appropriate functionality.

The point of control is a physical controller that reads some information from a card or QR code and checks by means of a request to the server whether the employee has access or not. A control point can have a parent point, for example:

- Polygon 1
    - Office 1001;
    - Office 1002;
- Polygon 2

There should be only one type of user in the system - this is an administrator.

The administrator's functionality should be as follows:

- Login to the system;
- Control points;
    - Creation of control points;
    - Viewing the list of control points;
- Adding employees to the system;
- Access groups;
    - Creation of access groups;
    - Viewing the list of access groups;
    - Adding a control point to an access group;
    - Removing a control point from an access group;
    - Adding an employee to an access group;
- Checking the employee's right to access the control point;
- View all access logs;
- Adding an employee, the right to access the control point for N seconds;

Authentication

Request for authentication in the system. When sending a request, you must send an object with a username and password. If the client sent the correct data, then it is necessary to return the generated token and username, otherwise an error message.

Request	Response

URL: {host}/api/login
Method: POST
 	
Headers
- Content-Type: application/json
 	
Body:
{
“login”: “admin”,
“password”: “admin”
}
 	
login - required parameter
password - required parameter
	---------- Successful ----------
	Status: 200
	Content-Type: application/json
	Body: {
	“data”: {
	“token”: <generated token>,
	“full_name”: “Alex Adm”
	}
	}

	---------- Validation error ----------
	Status: 422
	Content-Type: application/json
	Body: {
	“error”: {
	“code”: 422,
	“message”: “Validation error”,
	“errors”: {
	<key>: <error message>
	}
	}
	}
	
	---------- Unauthorized ----------
	Status: 401
	Content-Type: application/json
	Body: {
	“error”: {
	“code”: 401,
	“message”: “Unauthorized”,
	“errors”: {
	“login”: “invalid credentials”
	}
	}
	}

In all subsequent requests require authentication using the Authorization header**.** If the request is sent without this header or the token is invalid, then the server should return the following response in response:

Status: 401
Content-Type: application/json
Body: {
“error”: {
“code”: 401,
“message”: “Unauthorized”
}
}

Creating an employee

With the help of this request it should be possible to create an employee.

Request	Response

URL: {host}/api/staff
Method: POST
 	
Headers
- Content-Type: multipart/form-data
- Authorization: Bearer <token>
 	
Body:
- full_name: “Ivan Ivanov”
- photo: <photo file>
 	
full_name - required parameter
photo - required parameter, jpg
	---------- Successful ----------
	Status: 201
	Content-Type: application/json
	Body: {
	“data”: {
	“id”: 1,
	“full_name”: “Ivan Ivanov”,
	“code ”: <Unique employee code of 32 characters (generated randomly, assigned to the employee forever)>
	}
	}
	
	---------- Validation error ----------
	Status: 422
	Content-Type: application/json
	Body: {
	“error”: {
	“code”: 422,
	“message”: “Validation error”,
	“errors”: {
	<key>: < error message>
	}
	}
	}

Getting the list of employees

With the help of this request it should be possible to get the list of employees.

Request	Response

URL: {host}/api/staff
Method: GET
 	
Headers
- Authorization: Bearer <token>
	---------- Successful ----------
	Status: 200
	Content-Type: application/json
	Body: {
	“data”: {
	“items”: [
	{
	“id”: 1,
	“full_name”: “Ivan Ivanov”,
	“code”: <unique employee code of 32 characters>,
	“photo”: <direct link to the photo >
	}
	]
	}
	}

Creating a control point

A control point is a physical controller that checks an employee's access rights. This request should create a new control point in the system and return information about it. A control point can have a parent control point.

Request	Response

URL: {host}/api/points
Method: POST
 	
Headers
- Content-Type: application/json
- Authorization: Bearer <token>
 	
Body:
{
“name”: “Cluster A”,
“parent”: 1
}
 	
name - required parameter
parent - optional parameter, identifier of an existing control point
	---------- Successful ----------
	Status: 201
	Content-Type: application/json
	Body: {
	“data”: {
	“id”: 2,
	“name”: “Cluster A”,
	“parent”: <id / null>,
	}
	}
	---------- Validation error ----------
	Status: 422
	Content-Type: application/json
	Body: {
	“error”: {
	“code”: 422,
	“message”: “Validation error”,
	“errors”: {
	<key>: <error message>
	}
	}
	}

List of control points

This request should return all access points as nested form.

Request	Response

URL: {host}/api/points
Method: GET
 	
Headers:
- Authorization: Bearer <token>
	---------- Successful ----------
	Status: 200
	Content-Type: application/json
	Body: {
	"data": {
	"items": [
	{
	"id": 1,
	"name": "House 1",
	"points": [
	{
	"id": 2,
	"name": "Office 1",
	"points": []
	},
	{
	"id": 4,
	"name": "Office 2",
	"points": [
	{
	"id": 5,
	"name": "Chief office",
	"points": []
	}
	]
	}
	]
	},
	{
	"id": 6,
	"name": "House 2",
	"points": [
	{
	"id": 7,
	"name": "Office",
	"points": []
	}
	]
	}
	]
	}
	}

Creating an access group

An access group is a group that unites employees and allows you to provide access to control points for all members of the group at once.
This request should create a new access group and return information about it.

Request	Response

URL: {host}/api/groups
Method: POST
Headers:
- Content-Type: application/json
- Authorization: Bearer <token>
 	
Body: {
“name”: “Security”
}
 	
name - required parameter
	---------- Successful ----------
	Status: 201
	Content-Type: application/json
	Body: {
	“data”: {
	“id”: 1,
	“name”: “Security”
	}
	}
	---------- Validation error ----------
	Status: 422
	Content-Type: application/json
	Body: {
	“error”: {
	“code”: 422,
	“message”: “Validation error”,
	“errors”: {
	“name”: <error message>
	}
	}
	}

View all access groups

This query should return a list of all access groups.

Request	Response

URL: {host}/api/groups
Method: GET
 	
Headers:
- Authorization: Bearer <token>
	---------- Successful ----------
	Status: 200
	Content-Type: application/json
	Body: {
	“data”: {
	“items”: [
	{
	“id”: 1,
	“name”: “Security”
	},
	{
	“id”: 2,
	“name”: “Restaurant”
	}
	]
	}
	}

Adding control points to the access group

This request should add the control points sent in the request to the access group. In the body of the request, you must pass the points property with the IDs of the control points. The control point identifiers must be validated for existence.

Request	Response

URL: {host}/api/groups/<id>/points
Method: POST
 	
Headers
- Content-Type: application/json
- Authorization: Bearer <token>
 	
Body:
{
“points”: [1,2,3]
}
 	
points - required parameter, an array of existing identifiers to be added to the group
	---------- Successful ----------
	Status: 201
	---------- Validation error ----------
	Status: 422
	Content-Type: application/json
	Body: {
	“error”: {
	“code”: 422,
	“message”: “Validation error”,
	“errors”: {
	“points”: <error messages separated by commas>
	}
	}
	}

Adding an employee to an access group

This request must add the employees transferred in the request to the access group. In the body of the request, you must pass the staff property with employee IDs.
Employee IDs must be validated for existence.

Request	Response

URL: {host}/api/groups/<id>/staff
Method: POST
 	
Headers
- Content-Type: application/json
- Authorization: Bearer <token>
 	
Body:
{
“staff”: [1,2,3]
}
 	
staff - required parameter, an array of existing employee IDs to be added to the access group
	---------- Successful ----------
	Status: 201
	---------- Validation error ----------
	Status: 422
	Content-Type: application/json
	Body: {
	“error”: {
	“code”: 422,
	“message”: “Validation error”,
	“errors”: {
	“staff”: <Error messages separated by commas>
	}
	}
	}

Checking the employee's access rights

This request should check the employee's access rights to the control point. The request contains a unique employee code (not ID) and the ID of the control point, and the response returns the employee's photo and true / false, depending on the availability of access rights.

The right to access is determined by the following conditions:

- If the employee belongs to an access group that has access to the control point, then the employee must have access.
- If the above condition is not met, but the administrator has granted the employee the right to access within X seconds, then the employee must have access to the specified checkpoint for X seconds from the moment the access right was granted.

Since control points can have parent points, this means that if an employee is granted the right to access this point, then the employee should automatically have access to all parent points.

For example, let's imagine the following structure of points:

- Entrance
  - Polygon 1
    - Cabinet 1001;
    - Office 1002;
  - Polygon 2
    - Office 2001;

If an employee will have access to the control point "Cabinet 1002", then he must also have access to "Polygon 1" and "Entrance", respectively.

Any attempt to access must be logged in the system for further review.

Request	Response

URL: {host}/api/access
Method: POST
 	
Headers
- Content-Type: application/json
- Authorization: Bearer <token>
 	
Body:
{
“staff”: <unique 32-character employee code, not ID>,
“point”: <control point id>
}
 	
staff - required field, employee code
point - required field, existing control point ID
	---------- Successful ----------
	Status: 200
	Content-Type: application/json
	Body: {
	“data”: {
	“photo”: <link to employee photo>,
	“access”: <true / false>
	}
	}
	---------- Validation error ----------
	Status: 422
	Content-Type: application/json
	Body: {
	“error”: {
	“code”: 422,
	“message”: “Validation error”,
	“errors”: {
	<key>: <error message>
	}
	}
	}

Also, during the access check, you must save a link to the photo of the employee who initiated the access check. To do this, you need to refer to an external API and pass the access point ID there. In response, you will receive a link to a photo from a camera near this control point.

External API

Request	Response

URL: http://xkesryp-m1.wsr.ru/vcs/points/<ID>
Method: POST
 	
Query params:
- ID: checkpoint
	Status: 200
	Content-Type: application/json
	Body: {
	“data”: {
	“url”: <link to photo from camera>,
	}
	}

View all access logs

This request should return data about all attempts to access checkpoints.
Logs should be sorted by newest (newest first).

The ability to filter logs using theGET parameter must be implemented **type.** The parameter must take the following values:

- Parameter not specified - all logs must be returned.
- staff - the employee's logs should be returned (the employee's id is passed by the id GET parameter).
- point - the logs of the control point should be returned (the point id is passed by the id GET parameter).

Request	Response

URL: {host}/api/logs
Method: GET
 	
Headers
- Authorization: Bearer <token>
 	
Query params:
- type: staff | point
- id: existing employee or control point identifier (if the value of the parameter **type** is **staff** or **point**)
	Status: 200
	Content-Type: application/json
	Body: {
	“data”: {
	“items”: [
	{
	“access”: <true / false>,
	“staff”: {
	“id”: 1,
	“full_name”: “Alex”,
	“photo”: <link to profile photo>,
	“camera”: <link to photo from camera>
	},
	“point”: {
	“Id”: 1,
	“name”: “Security”
	},
	“timestamp”: 4239213213
	}
	]
	}
	}

Adding the employee, the right to access the control point for N seconds

With this request, the administrator must be able to grant the employee the right to access to the control point for the specified period of time.

Request	Response

URL: {host}/api/staff/<id>/access
Method: POST
 	
Headers
- Content-Type: application/json
- Authorization: Bearer <token>
 	
Body:
{
“point_id”: 7, // required field, existing ID of the control point
“time”: 3600 // required field, greater than 0, number of seconds for which access is granted
}
	---------- Successful ----------
	Status: 201
	
	---------- Validation error ----------
	Status: 422
	Content-Type: application/json
	Body: {
	“error”: {
	“code”: 422,
	“message”: “Validation error”,
	“errors”: {
	<key>: <error message>
	}
	}
	}

Database

You are provided with a ready-made database. You can add and remove data as you see fit, but you cannot change the structure of the database. When validating, all your data, **except for the users table,** will be replaced with other test data. Therefore, be sure to create a user with the appropriate username and password as described below.

INSTRUCTIONS TO THE COMPETITOR

The developed API must be available at http://xxxxxx-m1.wsr.ru/, where xxxxxx is the participant's login.

Formats for requests, responses, and date formats must match the examples in the assignment.

You must **definitely create an account user** in your system with the following login data:

- Login: admin
- Password: 1234

Only jobs uploaded to the server will be checked!

The works will be checked automatically!

EVALUATION SYSTEM

Section	Criterion	Amount
A	Work organization and management	4.00
B	Communication and interpersonal skills	4.00
C	Graphic design	0.00
D	Layout	0.00
E	Client-side	0.00
F	programmingServer-side programming	14.00
G	CMS	0.00
Total		22.00