worldskills Asia
Online Friendly Skills Game 2021

TEST PROJECT
for Web design and development

WSA2021_TP17_Client_side_actual_EN

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

CONTENTS

INTRODUCTION

Technologies of this module: JavaScript

Time to complete: 3 hours

The company "Smart control" has contacted you - the company is engaged in the organization of access control. They already have an API that implements access control, but there is no client part. You need to use all the available skills in client development to create a SPA.

The customer wants the client part to be easy to maintain, so the use of frameworks will be a plus.

DESCRIPTION OF PROJECT AND TASKS

Your task is to implement a SPA application that will work with an already developed API.
The application should be single-page and work without reloading the pages.
Your SPA should consist of the following screens:

- Registration screen
- Login screen
- Control point control screen
- Employee Management Screen
- Access Groups Management screen
- Access Group Management screen
- Logging screen

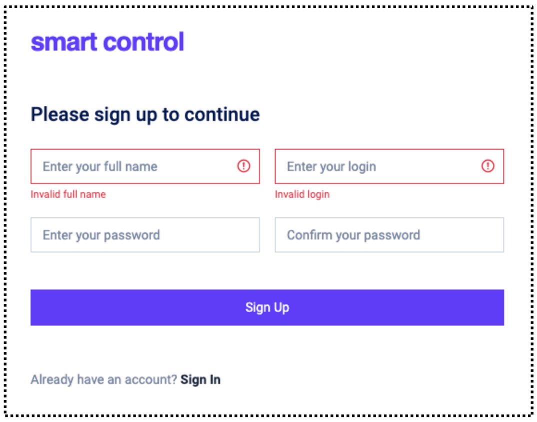
The customer provides you with a completely ready-made layout for all application screens. You can change the layout if you deem it necessary, but you should not remove data attributes from elements. The only data attribute whose value can be changed is the data-level attribute on the page with control points. At the same time, you can add and remove other attributes, as well as change the nesting of the layout, but the finished application should not be modified.

There are errors related to validation and usability in the template that is provided to you. You need to fix them. At the same time, validation errors related to the use of the framework will not be taken into account during verification.

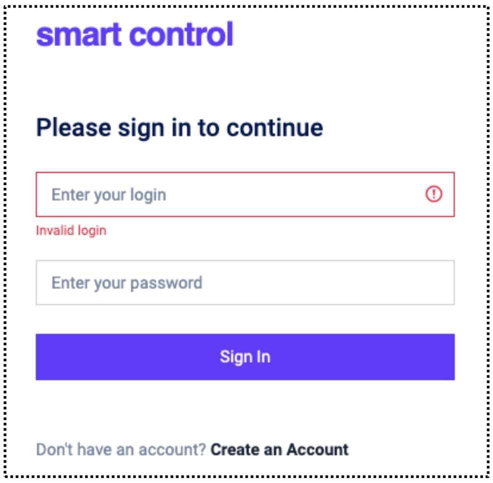
Registration screen

In the signup file.html you are provided with the layout of this screen. The screen contains a registration form, as shown in the screenshot.

1. When you click on the "Sign Up" button, a request should be sent to the API to register a new user.
2. In case of a successful response from the server, the user should be redirected to the login screen.
3. In case of errors, it is necessary to display errors from the server under the corresponding input fields.
4. When you click on the "Sign In" link, the screen should change to the login screen.



Login screen



smart control

Please sign in to continue

Enter your login

Invalid login

Enter your password

Sign In

Don't have an account? **Create an Account**

In file signin.html you are provided with the layout of this screen. The screen contains a login form, as shown in the screenshot.

1. If the user has not yet logged in, this screen should open when going to the site.

2. When you click on the "Sign In" button, a request should be sent to the API to verify the entered data.

3. In case of a successful response from the server, the user should be remembered and the control point management screen should be shown to him.

4. If there are errors in the response, they must be displayed under the corresponding input fields.

5. When you click on the link "Create an Account", the screen should change to the signup screen.

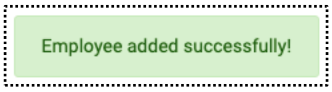
Application header

1. When clicking on the link in the navigation, the user should see the corresponding screen.
2. When you click on the avatar, a submenu with user information should be shown.
3. When you click on the avatar again, the submenu in the header should be hidden.
4. When you click on the "Sign Out" link, you should log out and the user should be shown the login screen.

Modal Windows

In the application, you will need to implement a number of modal windows. The user should be able to close the modal window by clicking on the button with a cross. This function should work for all modal windows.

Success Messages



Employee added successfully!

When performing actions related to creating, updating and deleting data, it is necessary to display success messages in the form of pop-up notifications. Notifications should appear in the lower right corner of the screen immediately after performing the action and disappear after 3 seconds.

An example of the design of notifications can be found in the file notification-example.html.

The table below shows the text that should be displayed in notifications according to the action.

Action	**Notification text**
Adding a checkpoint	Control point added successfully!
Changing the control point	Control point updated successfully!
Deleting a checkpoint	Control point deleted successfully!

Adding an employee	Employee added successfully!
Adding an access group	The access group has been successfully created!
Deleting an access group	Access group deleted successfully!

Animation

The customer wants the web application to have as much animation as possible. Use your knowledge and skills to add animation to the application.

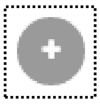
Control point management screen

In the file index.html you are provided with the layout of this screen. When you click on the "Control point management" link in the header, the user should appear on this screen.

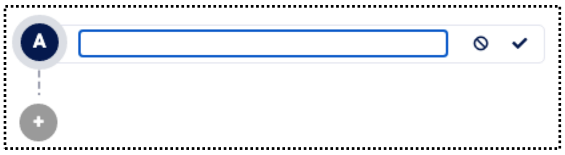
If the user has already logged in, this screen should open when the page is refreshed.

This screen should display all control points received from the API.

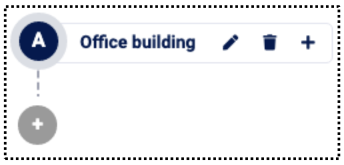
If the user does not have any control points yet, then initially only the button for adding a new control point should be displayed (as in the figure on the right).



When you click on the add button, the root level point should be added to the structure as shown in the picture on the right. This point should have a field for entering the name, a cancel button and an application button.



When you click on the cancel button, this point should be deleted, and when you click on the apply button, a request to the API for creating a point should occur. In case of successful creation, the point should take the following form.

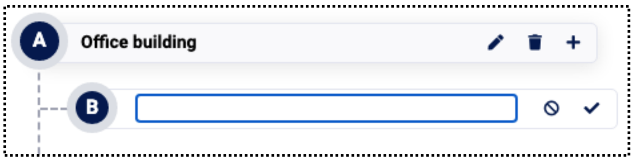


The created point should have 3 functional buttons: edit, delete, add a nested point.

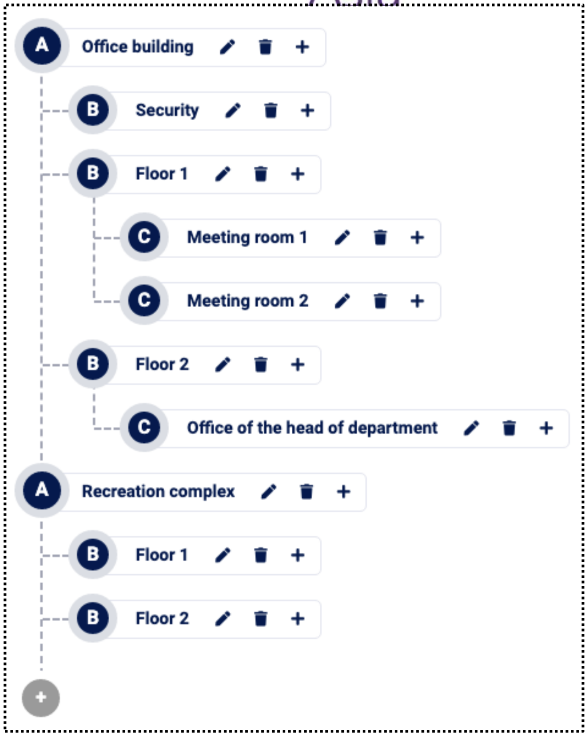
When you click on the button with the edit icon, the control point should again take the form as when it was created to change the entered information. After changing the name, you need to click on the cancel icon to not save the data, or on the confirmation icon to save the new name. The changed name must be saved in the API.

When the delete button is clicked, an API request should be made to delete the corresponding point, and if the response is successful, the point and all its child points should be removed from the page.

When you click on the button to add a nested point, a new nested point should be displayed on the screen as in the picture on the right.



There may be several root points and as many nested ones as you want. The picture on the right shows one of the possible structures.



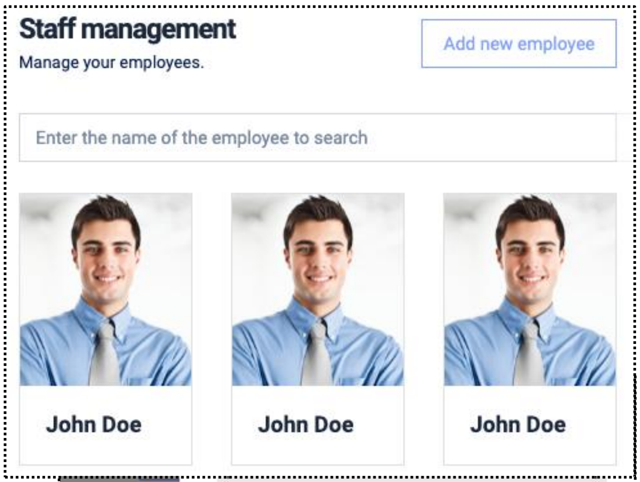
Each nesting level should have its own letter of the English alphabet in uppercase: ABCDEFGHIJKLMNOPQRSTUVWXYZ. This is already taken into account in the layout using the data-level="X " attribute, where you need to specify the desired letter instead of X. This data attribute can and should be changed.

After the last root point, a button should always be displayed to add another root point.

You do not need to make up any of the above yourself. Carefully study the layout provided to you and implement the required functionality.

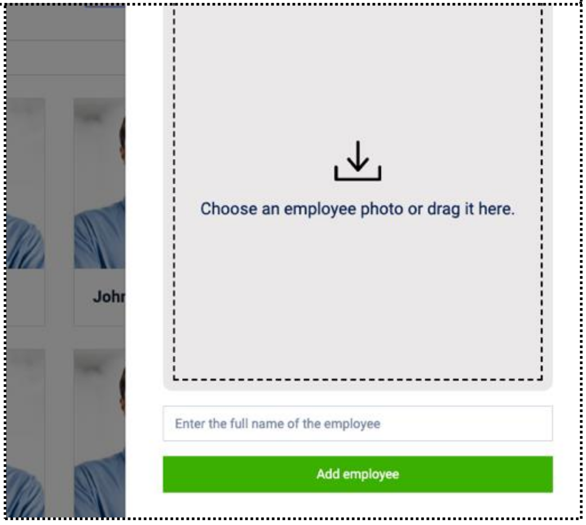
Employee Management Screen

In the file staff.html you are provided with the layout of this screen. When you click on the "Staff management" link in the header, the user should appear on this screen. You need to display employees on the screen, based on data from the API.



Also on this screen there should be a search by the full name of the employee. Cards that do not meet the condition must be hidden. If the input field is empty, then all the cards should be displayed. The search must be case-insensitive and implemented on the client side.

When clicking on the "Add new employee" button, the user should see a modal window on the right side of the screen, as in the picture on the right.

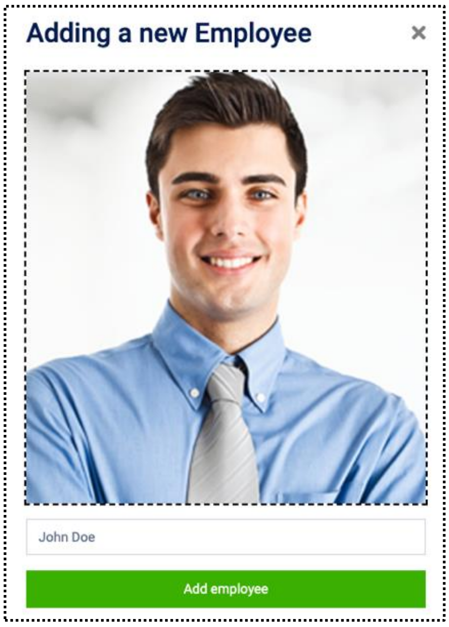

Choose an employee photo or drag it here.
Enter the full name of the employee
Add employee

The user should be able to select a file with an employee's photo by clicking on the gray area in the modal window or dragging the file there.

The selected or dragged photo file should be displayed instead of the gray area, as in the picture on the right.

To do this, display the img element with the data-staff-form-preview attribute.

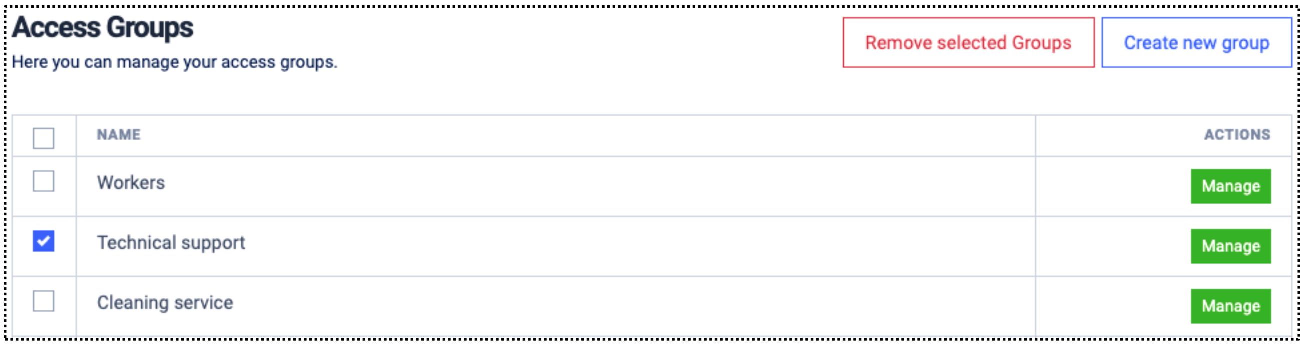
When you click on the "Add employee" button, a request should be sent to the API to create a new employee, and if the response is successful, the added employee should be displayed in the general list, and the modal window should be closed.


Adding a new Employee ×

John Doe

Access Groups Management Screen

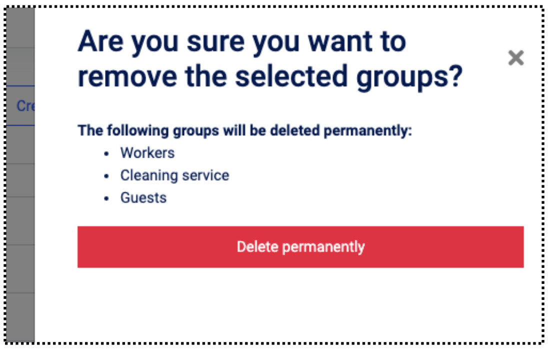
In the file access.html you are provided with the layout of this screen. When you click on the "Access groups" link in the header, the user should appear on this screen.


Access Groups

Here you can manage your access groups.

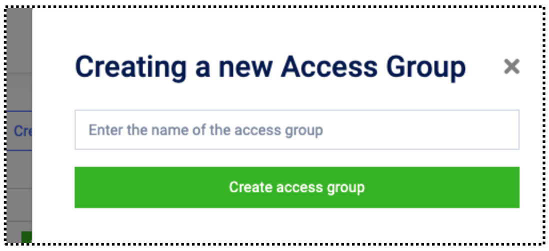
☐	NAME	ACTIONS
☐	Workers	Manage
✓	Technical support	Manage
☐	Cleaning service	Manage

1. The table should display all access groups received from the API.
2. It should be possible to select all or individual groups.
3. When you click on the checkbox in the table header, all the rows of the table should be highlighted.
4. The "Remove selected Groups" button should be displayed only if at least one row is selected.
5. When you click on the "Remove selected Groups" button, a modal window should be displayed asking you to confirm the deletion of the selected groups. The selected groups should be listed as a list.
6. When you click the "Delete permanently" button in the modal window, the selected groups should be deleted from the server using the API, the modal window should be closed and the deleted groups should not be in the table.


Are you sure you want to remove the selected groups? ×

The following groups will be deleted permanently:

- Workers
- Cleaning service
- Guests



Creating a new Access Group ×

Enter the name of the access group

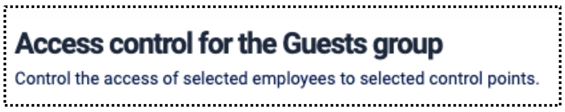
Create access group

When clicking on the "Create new group" button, the user should see a modal window on the right side of the screen, as in the picture on the right.

When you click on the "Create access group" button, a request should be sent to the API to create a new access group, and if the response is successful, the added group should be displayed in the group table, and the modal window should be closed.

Access Group Management Screen

In the file access-manage.html you are provided with the layout of this screen. The transition to this screen should be carried out by clicking on the "Manage" button in the access group.



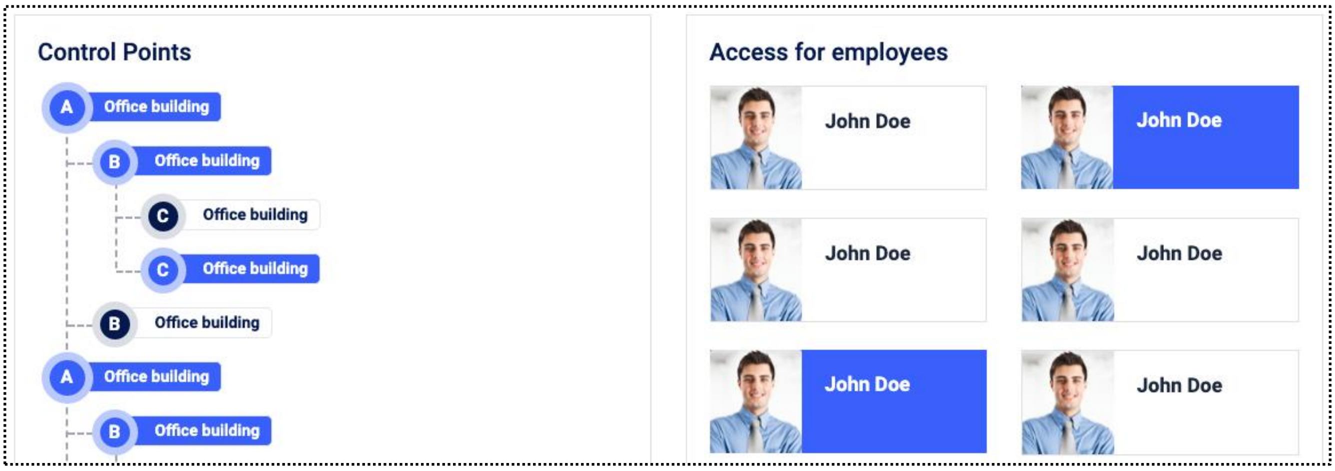
Access control for the Guests group

Control the access of selected employees to selected control points.

The name of the screen should include the name of the selected access group, as in the picture on the right.
Guests is the name of the group

Also, on this screen, you need to display the existing structure of control points and a list of all employees.

Checkpoints and employees that are included in the access group should be highlighted, as in the picture below.



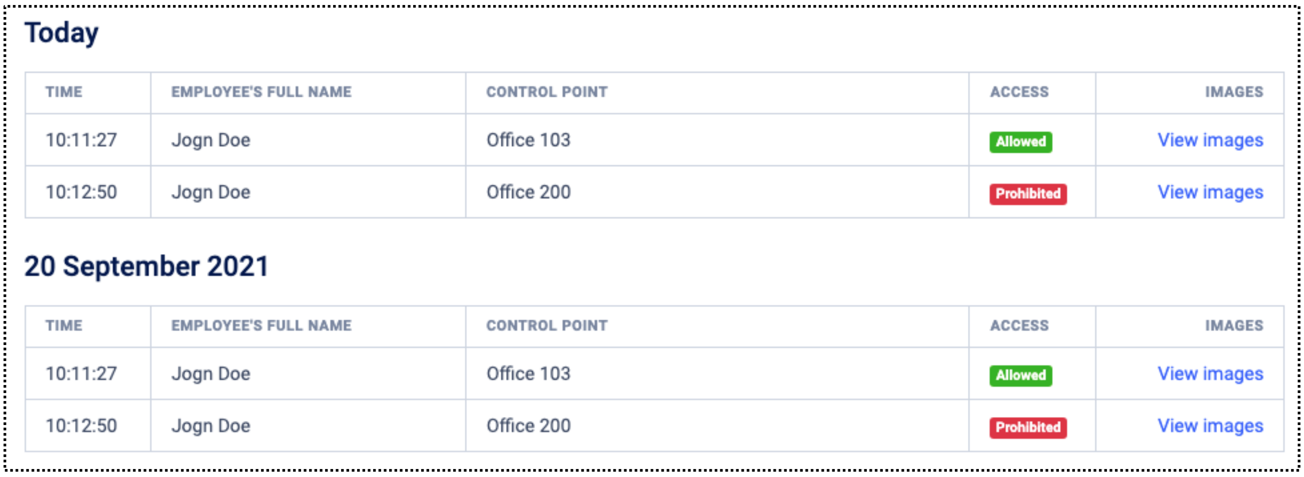
When you click on the control point, an API request should be made to add the selected point to the current group. If the response is successful, the dot should stand out.

When you click on the selected point again, which is included in the access group, an API request should be made to remove the selected point from the access group. If the response is successful, the point should no longer be highlighted.

Actions when clicking or clicking on employees again are similar to the above.

Logging screen

In the file logging.html you are provided with the layout of this screen. When you click on the "Logging and analytics" link in the header, the user should appear on this screen. The screen should display all access logs received from the API, as in the screenshot below.



Today

TIME	EMPLOYEE'S FULL NAME	CONTROL POINT	ACCESS	IMAGES
10:11:27	Jogn Doe	Office 103	Allowed	View images
10:12:50	Jogn Doe	Office 200	Prohibited	View images

20 September 2021

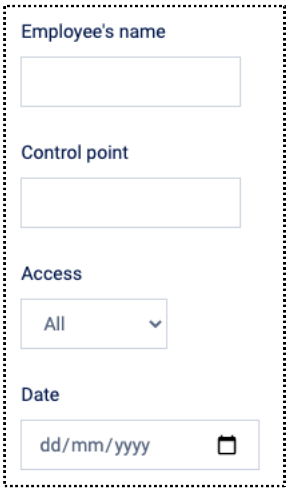
TIME	EMPLOYEE'S FULL NAME	CONTROL POINT	ACCESS	IMAGES
10:11:27	Jogn Doe	Office 103	Allowed	View images
10:12:50	Jogn Doe	Office 200	Prohibited	View images

Logs should be grouped by date and sorted from new to old. At the same time, within the grouped date, the logs should be sorted from old to new.

Logs for today should be in the Today group, and all the others in the format "DD Month YYYY"

Each line with the log should display

- Time – in the HH format:MM:SS
- Full name of the employee
- Name of the control point
- Access – if access was allowed (true), the green label "Allowed" should be displayed, otherwise the red one "Prohibited".
- Link to view images – this link should not lead anywhere and open anything.



The filtering settings are located on the right side of the page. You need to implement the following filtering on the client side:

- Case - insensitive filtering by employee name
- Filtering by the name of the control point without taking into account the case
- Filtering by access – All/Allowed/Prohibited
- Filtering by date

All fields should simultaneously affect filtering. If the field is not filled in, then filtering by this field is not taken into account. Filters should be applied immediately after changing the field or entering a new character.

INSTRUCTIONS TO THE COMPETITOR

The developed application must be available at http://xxxxxx-m3.wsr.ru/, where xxxxxx is the username of the participant.

The work will be checked in Google Chrome.

Only works uploaded to the server are checked.

If you are using the CLI version of the framework, then place your source code files (not compressed) in the source folder on the server for the possibility of evaluating your code.

Do not upload node_modules to the server!

EVALUATION SYSTEM

section	Criterion	Sum
A	Organization of work and management	2,00
B	Communication and interpersonal skills	2,00
C	Graphic Design	4,00
D	Layout	7,00
E	Client-side programming	17,00
F	Server-side programming	0,00
G	CMS	0,00
Total		**32,00**

API Documentation

The alias {host} denotes the address http://megpfjd-m3.wsr.ru

Request for registration

URL: {host}/api/register
Method: POST
	Headers
	- Content-Type: application/json
		Body: {
		“full_name”: “Ivan Ivanov”,
		“login”: “ivan2021”,
		“password”: “my-pass”
		}

--------------- Successful response -------------
Status: 201
Content-Type: application/json
Body: {
“data”: {
“id”: 1,
“full_name”: “Ivan Ivanov”
}
}
	---------------------- Validation error response --------------------
	Status: 422
	Content-Type: application/json
	Body: {
	“error”: {
	“code”: 422,
	“message”: “Validation error”,
	“errors”: { <key>: <error message> }
	}
	}

Request for authentication

URL: {host}/api/login
Method: POST
	Headers
	- Content-Type: application/json
		Body: {
		“login”: “ivan2021”,
		“password”: “my-pass”
		}

------ Successful response -----
Status: 200
Content-Type: application/json
Body: {
“data”: {
“token”: <token>,
“full_name”: “Ivan Ivanov”
}
}
	--- Validation error response --
	Status: 422
	Content-Type: application/json
	Body: {
	“error”: {
	“code”: 422,
	“message”: “Validation error”,
	“errors”: {
	<key>: <message>
	}
	}
	}
		-------- Unauthorized response --------
		Status: 401
		Content-Type: application/json
		Body: {
		“error”: {
		“code”: 401,
		“message”: “Unauthorized”,
		“errors”: {
		“login”: “invalid credentials”
		}
		}
		}

Request to get all access points

Request	Response

URL: {host}/api/points
Method: GET
 	
Headers:
- Authorization: Bearer <token>
	Status: 200
	Content-Type: application/json
	Body: {
	  "data": {
	    "items": [{
	      "id": 1,
	      "name": "Department A",
	      "points": [{
	          "id": 2,
	          "name": "Office A1",
	          "points": []
	      }]
	    }]
	  }
	}

Request to create a new access point

Request	Response

URL: {host}/api/points
Method: POST
 	
Headers
- Content-Type: application/json
- Authorization: Bearer <token>
 	
Body: {
  "name": "Office A2",
  "parent": 1
}
	Status: 201
	Content-Type: application/json
	
	Body: {
	  "data": {
	    "id": 2,
	    "name": "Office A2",
	    "parent": 1,
	  }
	}

Request to edit an access point

Request	Response

URL: {host}/api/points/<ID>
Method: PATCH
 	
Headers
- Content-Type: application/json
- Authorization: Bearer <token>
 	
Body: {
  "name": "Office A2-1"
}
	Status: 200
	Content-Type: application/json
	
	Body: {
	  "data": {
	    "name": "Office A2-1"
	  }
	}

Request to delete an access point

Request	Response
URL: {host}/api/points/<ID> **Method:** DELETE **Headers** **- Authorization:** Bearer <token>	**Status:** 200

Request to get all groups

Request	Response

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

Request to create a new access group

Request	Response

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

Request to delete an access group

Request	Response
URL: {host}/api/groups/<ID> **Method:** DELETE **Headers** **- Authorization:** Bearer <token>	**Status:** 200

Request to get an access group

Request	Response

URL: {host}/api/groups/<ID>
Method: GET
 	
Headers
- Authorization: Bearer <token>
	Status: 200
	Content-Type: application/json
	Body: {
	  “data”: {
	    “id”: 1,
	    “name”: “Office A”,
	    “staff”: [1, 5, 6],
	    “points”: [4, 5, 8]
	  }
	}

Request to add access points to a group

Request	Response

URL: {host}/api/groups/<ID>/points
Method: POST
 	
Headers
- Authorization: Bearer <token>
 	
Body {
  “points”**:** [1, 4]
}
	Status: 201
	-------------------- Validation error response -------------------
	Status: 422
	Content-Type: application/json
	Body: {
	  “error”: {
	    “code”: 422,
	    “message”: “Validation error”,
	    “errors”: {
	      “points”: <error messages separated by commas>
	    }
	  }
	}

Request to remove access points from a group

Request	Response

URL: {host}/api/groups/<ID>/points
Method: DELETE
 	
Headers
- Authorization: Bearer <token>
 	
Body {
“points”**:** [1, 4]
}
	Status: 201
	
	-------------------- Validation error response -------------------
	Status: 422
	Content-Type: application/json
	Body: {
	“error”: {
	“code”: 422,
	“message”: “Validation error”,
	“errors”: {
	“points”: <error messages separated by commas>
	}
	}
	}

Creating 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: <file>
 	
full_name – required
photo – required, jpg
	--------------------- Successful --------------------
	Status: 201
	Content-Type: application/json
	Body: {
	“data”: {
	“id”: 1,
	“full_name”: “Ivan Ivanov”,
	“code”: <generated code>
	“photo”: <photo link>
	}
	}
	------------------ Validation error ----------------
	Status: 422
	Content-Type: application/json
	Body: {
	“error”: {
	“code”: 422,
	“message”: “Validation error”,
	“errors”: {
	<key>: <error message>
	}
	}
	}

worldskills
Asia

Getting a 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”: <code>,
	        “photo”: <photo link>
	      }
	    ]
	  }
	}

Request to add employees to a group

Request	Response

URL: {host}/api/groups/<ID>/staff
Method: POST
 	
Headers
- Authorization: Bearer <token>
 	
Body {
  “staff”**:** [6]
}
	Status: 201
	------------------- Validation error response ------------------
	Status: 422
	Content-Type: application/json
	Body: {
	  “error”: {
	    “code”: 422,
	    “message”: “Validation error”,
	    “errors”: {
	      “staff”: <error messages separated by commas>
	    }
	  }
	}

Request to remove employees from the group

Request	Response

URL: {host}/api/groups/<ID>/staff
Method: DELETE
 	
Headers
- Authorization: Bearer <token>
 	
Body {
  “staff”**:** [6]
}
	Status: 201
	-------------------- Validation error response -------------------
	Status: 422
	Content-Type: application/json
	Body: {
	  “error”: {
	    “code”: 422,
	    “message”: “Validation error”,
	    “errors”: {
	      “staff”: <error messages separated by commas>
	    }
	  }
	}

Request for getting logs

Request	Response

URL: {host}/api/logs
Method: GET
 	
Headers
- Authorization: Bearer <token>
	Status: 200
	Content-Type: application/json
	Body: {
	  “data”: {
	    “items”: [
	      {
	        “access”: <true/false>,
	        “staff”: “Alex”,
	        “point”: “Office A1”,
	        “timestamp”: 4239213213
	      }
	    ]
	  }
	}