**Code:** [**https://github.com/RamazTs/ProjectBackend**](https://github.com/RamazTs/ProjectBackend) **API documentation and Endpoints:**

General Notes:

* Endpoints that have the path prefix “**public**” do not require a JWT token to access, otherwise, a token needs to be provided.
* To provide the token to the endpoint, add “**Authorization**” header to the request and add the JWT token to its value
* For dynamic route “:userId” please provide user id associated with the account you are logged in as.
* If there’s an error from the endpoints the general format of the error object is as follows:

{

"error": "ERROR\_EMAIL\_ALREADY\_EXISTS"

}

Object with the error key and the value displaying the reason for the error

User Endpoints:

/api/public/register

Method: POST

{

"name": required,

"lastname": required,

"password": required,

"email" : required,

"occupation": required

}

Returns the created user as a response if successful.

VALIDATION Regular expressions:

Name: "^(?=.\*[\\p{L}].\*[\\p{L}]).{2,}$"

Email: "^[a**-**zA**-**Z0**-**9][a**-**zA**-**Z0**-**9.!#$%&'\*+\\/=?^\_`{|}~-]\*@[a**-**zA**-**Z0**-**9-]{2,}(\\.[a**-**zA**-**Z0**-**9]{2,})+$

Occupation: Either “Professor” or “Student”

Password:

At least one upper case: "[A**-**Z]+"

At least one lower case: "[a**-**z]+"

At least one digit: "\\d+"

Length must be between 8 – 20 characters: "^.{8,20}$"

Response Ex:

{

"\_id": "USER-8c620c62-9efa-4211-ab3b-cd910eb91ce5",

"name": "TestName",

"lastname": "TestLastName",

"email": "test.email@email.coms",

"occupation": "STUDENT",

"registered\_at": "2024-04-17T04:24:28.973Z"

}

/api/public/login

Method: POST

{

"password": required,

"email" : required

}

Returns JWT token and userId if successful

Response Ex:

"jwt": "eyJhbGciOiJIUzUxMiIsInR5cCI6IkpXVCJ9.eyJ1c2VyIjp7Il9pZCI6IlVTRVItNzAwNGJiNjAtZmM2OS00MmVkLTllZDMtNzkyYjA5NTJmODllIiwibmFtZSI6IlRlc3ROYW1lIiwibGFzdG5hbWUiOiJUZXN0TGFzdE5hbWUiLCJwYXNzd29yZCI6IiQyYSQxMiRrY1RhV1pVMHlEY3lnLlNHejRZemEuSmFPVlBOT1IwTDNycW4uMk9sYkVNYlJzZlJEZlUvaSIsImVtYWlsIjoidGVzdC5lbWFpbEBlbWFpbC5jb20iLCJvY2N1cGF0aW9uIjoiU1RVREVOVCIsInJlZ2lzdGVyZWRfYXQiOiIyMDI0LTA0LTE3VDA0OjA3OjIzLjQ0NFoifSwiZXhwaXJlc0luIjo5MDAsImlhdCI6MTcxMzMyNjg5OX0.hPd-DpWsK4YDxTdOCdmOjgnsCe8AqkUl-z6ZMkClrcYPRuCNVX4MP1Bwkrl9xXUiMX-1k3pe1bhD60jnOSkjeQ",

"userId": "USER-7004bb60-fc69-42ed-9ed3-792b0952f89e"

/api/:userId

Method: GET, requires Auth

Does not require a request body.

Response Ex:

{

"\_id": "USER-7004bb60-fc69-42ed-9ed3-792b0952f89e",

"name": "TestName",

"lastname": "TestLastName",

"email": "test.email@email.com",

"occupation": "STUDENT",

"registered\_at": "2024-04-17T04:07:23.444Z"

}

/api/professors

Method: GET, requires Auth

Gets all of the professors registered on the platform.

No request body required.

{

"professors": [

{

"name": "TestProfessorName",

"lastname": "ProfessorTestLastName",

"email": "professor.email@email.coms",

"\_id": "USER-e8956aa2-fed8-4b5e-9b40-3c82ea4957fd"

}

]

}

Schedules:

/api/schedule/:professorId

Method: GET, requires Auth

Returns the schedule for the professor and the students who have registered to the professor’s schedule

Example response:

{

"professorItems": [

{

"\_id": "PROFESSOR-ITEM-012a960e-61bd-423e-a6c9-c09712189e92",

"parent\_id": "PROFESSOR-ITEM-PARENT-da0439f8-8963-4243-920a-ae749819965f",

"professor\_id": "USER-e754dec1-1674-4a86-b924-2614b7280693",

"itemName": "some\_other\_name",

"startTime": "2024-04-24T00:29:26Z",

"endTime": "2024-04-24T01:29:26Z",

"date\_created": "2024-04-24T01:43:02.863Z",

"active": "ACTIVE",

"recurring": false

},

{

"\_id": "PROFESSOR-ITEM-f7f0ca40-10cb-453c-a5c0-8f7de04176a2",

"parent\_id": "PROFESSOR-ITEM-PARENT-da0439f8-8963-4243-920a-ae749819965f",

"professor\_id": "USER-e754dec1-1674-4a86-b924-2614b7280693",

"itemName": "some\_other\_name",

"startTime": "2024-05-01T00:29:26.000Z",

"endTime": "2024-05-01T01:29:26.000Z",

"date\_created": "2024-04-24T01:43:02.863Z",

"active": "ACTIVE",

"recurring": false

},

{

"\_id": "PROFESSOR-ITEM-ee0710f2-026c-4386-bc9c-0baf4cd3070d",

"parent\_id": "PROFESSOR-ITEM-PARENT-da0439f8-8963-4243-920a-ae749819965f",

"professor\_id": "USER-e754dec1-1674-4a86-b924-2614b7280693",

"itemName": "some\_other\_name",

"startTime": "2024-05-08T00:29:26.000Z",

"endTime": "2024-05-08T01:29:26.000Z",

"date\_created": "2024-04-24T01:43:02.863Z",

"active": "ACTIVE",

"recurring": false

},

{

"\_id": "PROFESSOR-ITEM-e6567d21-0eff-468f-9549-3df58da91f0a",

"parent\_id": "PROFESSOR-ITEM-PARENT-da0439f8-8963-4243-920a-ae749819965f",

"professor\_id": "USER-e754dec1-1674-4a86-b924-2614b7280693",

"itemName": "some\_other\_name",

"startTime": "2024-05-15T00:29:26.000Z",

"endTime": "2024-05-15T01:29:26.000Z",

"date\_created": "2024-04-24T01:43:02.863Z",

"active": "ACTIVE",

"recurring": false

},

{

"\_id": "PROFESSOR-ITEM-9e4be8ab-e950-4421-86d8-9e3756f6e4d2",

"parent\_id": "PROFESSOR-ITEM-PARENT-da0439f8-8963-4243-920a-ae749819965f",

"professor\_id": "USER-e754dec1-1674-4a86-b924-2614b7280693",

"itemName": "some\_other\_name",

"startTime": "2024-05-22T00:29:26.000Z",

"endTime": "2024-05-22T01:29:26.000Z",

"date\_created": "2024-04-24T01:43:02.863Z",

"active": "ACTIVE",

"recurring": false

}

],

"studentItems": [

{

"\_id": "STUDENT-ITEM-e9244a90-f2bb-4856-adf6-2e3b399214fb",

"professor\_id": "USER-e754dec1-1674-4a86-b924-2614b7280693",

"professor\_item\_id": "PROFESSOR-ITEM-012a960e-61bd-423e-a6c9-c09712189e92",

"student\_id": "USER-5ec06c0a-2fd2-4316-b442-e6812fec6e27",

"student\_name": "TestName",

"student\_last\_name": "TestLastName",

"student\_email": "test.email@email.coms",

"startTime": "2024-04-24T01:11:26.000Z",

"endTime": "2024-04-24T01:20:26.000Z",

"date\_created": "2024-04-24T01:43:22.954Z",

"active": "ACTIVE"

}

]

}

/api/professor/schedule/:userId

Method: Post, requires Auth, needs to be a professor

Professor creates schedule entries which provide windows for students to enter their meeting times.

JSON Body Requirements:

Times need to be provided in an ISO string format for example: “2024-04-24T00:29:26Z”

Item name: string

Recurring: Boolean (if recurring is true, the app will repeat this scheduled item for the next 4 weeks on the same day and time)

{

"item\_name": required,

"start\_time": required

"end\_time": required,

"recurring": required

}

JSON body example:

{

"item\_name": "some\_other\_name",

"start\_time": "2024-04-24T00:29:26Z",

"end\_time": "2024-04-24T01:29:26Z",

"recurring": true

}

/api/professor/schedule/:userId

Method: Delete, requires Auth, needs to be a professor

Deletes a single schedule entry.

JSON Body Requirements:

ItemId: string

Example:

{

"itemId" : "PROFESSOR-ITEM-5bc6a7a4-9b56-4828-a408-a4f4ad78014b"

}

Example Success Response:

{

"parent\_id": "PROFESSOR-ITEM-PARENT-7f5839c3-8336-48ee-8794-b90a76aa8c36",

"status": "Success"

}

api/professor/schedule/recurring/:userId

Method: Delete, requires Auth, needs to be a professor

Deletes all schedule entries belonging to a recurring meeting.

JSON Body Requirements:

ItemParentId: string

Example:

{

"itemParentId": "PROFESSOR-ITEM-PARENT-7f5839c3-8336-48ee-8794-b90a76aa8c36"

}

/api/student/schedule/:userId

Method: Post, requires Auth

Student can add his/her item to an existing time window provided by the professor’s schedule.

JSON Body Requirements:

Times need to be provided as ISO string

startTime: string,

endTime: string

proessorItemId: string

{

"professorItemId": required

"startTime": required

"endTime": required

}

Example Request

{

"professorItemId": "PROFESSOR-ITEM-012a960e-61bd-423e-a6c9-c09712189e92",

"startTime": "2024-04-24T01:11:26Z",

"endTime": "2024-04-24T01:20:26Z"

}

Example Response:

{

"\_id": "STUDENT-ITEM-e9244a90-f2bb-4856-adf6-2e3b399214fb",

"professor\_id": "USER-e754dec1-1674-4a86-b924-2614b7280693",

"professor\_item\_id": "PROFESSOR-ITEM-012a960e-61bd-423e-a6c9-c09712189e92",

"student\_id": "USER-5ec06c0a-2fd2-4316-b442-e6812fec6e27",

"student\_name": "TestName",

"student\_last\_name": "TestLastName",

"student\_email": "test.email@email.coms",

"startTime": "2024-04-24T01:11:26.000Z",

"endTime": "2024-04-24T01:20:26.000Z",

"date\_created": "2024-04-24T01:43:22.954Z",

"active": "ACTIVE"

}

/api/student/schedule/:userId

Method: Delete, requires Auth

Deletes student schedule item, professors can also delete student items.

JSON requirement:

ItemId: string

{

"itemId": required

}

Example:

{

"itemId": "STUDENT-ITEM-e9244a90-f2bb-4856-adf6-2e3b399214fb"

}

Example response:

{

"\_id": "STUDENT-ITEM-e9244a90-f2bb-4856-adf6-2e3b399214fb",

"status": "Success"

}