API-REST

Group 6

Login

- **Login Message:**
  - **BACK-END:**
    - URL: “/api/sessions”
    - Method: POST
    - Type of parameter: JSON
    - Parameter: {email:<string> , password:<string>}
      - email has got correct syntax of a email address.
      - password is not null and has got at least 8 characters.
  - **FRONT-END**
    - Success Response:
      - Type of response: JSON
        (When server process the request, use status code 200)
      1. Data in response: {
         success: <message>,
         email: <string>,
         password: <string>,
         name: <string>,
         surname: <string>,
         gender: <string>,
         role : <string>,
         changepwd : <string>
         }
         - success message can be:
           - “User logged in.”
         - gender only can be:
           - “male”
           - “female”
         - role only can be:
           - “Developer"
           - "Project Manager"
           - "Quality Assurance"
           - "Business Analyst"
           - "Other"
         - changepwd only can be:
           - “true”
           - “false”
      2. Data in response:{
         error : <message>
         }
         - error message can be:
           - "Username or password incorrect."
    - Error Response:
      - Type of response: JSON
        (When server can’t process the request, use status code 4xx)
      1. Data in response:{
         error : <message>
         }
         - error message can be:
           - "Incorrect request."

- **Logout Message:**
    - **BACK-END:**
        - URL: “/api/sessions/logout”
        - Method: POST
        - Type of parameter: JSON
        - Parameter: none
    - **FRONT-END**
        - Success Response:
            - Type of response: JSON
              (When server process the request, use status code 200)
            1. Data in response: {
               success: <message>
               }
                - success message can be:
                    - “User logged out.”
        - Error Response:
            - Type of response: JSON
              (When server can’t process the request, use status code 4xx)
            1. Data in response:{
               error : <message>
               }
                - error message can be:
                    - "Incorrect request."
- **Forgot Password Message:**
    - **BACK-END:**
        - URL: “/api/sessions/forgotpassword”
        - Method: POST
        - Type of parameter: JSON
        - Parameter: {email:<string>}
            - email has got correct syntax of a email address.
    - **FRONT-END**
        - Success Response:
            - Type of response: JSON
              (When server process the request, use status code 200)
            1. Data in response: {
               success: <message>
               }
                - success message can be:
                    - “Sent an email to “ + user_email + “ with a random password.”
            2. Data in response:{
               error : <message>
               }
                - error message can be:
                    - "This email not is a username.."
        - Error Response:
            - Type of response: JSON
              (When server can’t process the request, use status code 4xx)
            1. Data in response:{
               error : <message>
               }
                - error message can be:
                    - "Incorrect request."

Users

- **Create User Message:**
  - **BACK-END:**
    - URL: “/api/users”
    - Method: POST
    - Type of parameter: JSON
    - Parameter: {email:<string>, password:<string>, name:<string>, surname:<string>, gender<string>, role:<string>, photo:<string>}
      - email has got correct syntax of a email address.
      - password is not null and has got at least 8 characters.
      - name is not null.
      - surname is not null.
      - gender is not null.
      - photo is a file.
  - **FRONT-END**
    - Success Response:
      - Type of response: JSON
        (When server process the request, use status code 200)
      1. Data in response: {
         success: <message>
         }
         - success message can be:
           - “Added new user " + user_email + " to the database.”
      2. Data in response:{
         error : <message>
         }
         - error message can be:
           - “Email not valid.”
           - "Mail already present into the database."
           - "Name or surname not valid."
           - "Password not valid."
           - "Role or photo not valid."
    - Error Response:
      - Type of response: JSON
        (When server can’t process the request, use status code 4xx)
      1. Data in response:{
         error : <message>
         }
         - error message can be:
           - "Incorrect request."

- **Delete User Message:**
    - **BACK-END:**
        - URL: “/api/users/delete”
        - Method: POST
        - Type of parameter: JSON
        - Parameter: none
    - **FRONT-END**
        - Success Response:
            - Type of response: JSON
              (When server process the request, use status code 200)
            1. Data in response: {
               success: <message>
               }
                - success message can be:
                    - “Removed user " + user_email + " from the database.”
            2. Data in response:{
               error : <message>
               }
                - error message can be:
                    - “User not logged in.”
                    - "Mail not present into the database."
        - Error Response:
            - Type of response: JSON
              (When server can’t process the request, use status code 4xx)
            1. Data in response:{
               error : <message>
               }
                - error message can be:
                    - "Incorrect request."

- **Change User Password Message:**
    - **BACK-END:**
        - URL: “/api/users/changepassword”
        - Method: POST
        - Type of parameter: JSON
        - Parameter: {email:<string>, new_password:<string>}
            - email has got correct syntax of a email address.
            - password is not null and has got at least 8 characters.
    - **FRONT-END**
        - Success Response:
            - Type of response: JSON
              (When server process the request, use status code 200)
            2. Data in response: {
               success: <message>
               }
                - success message can be:
                    - “Password successfully changed.”
            3. Data in response:{
               error : <message>
               }
                - error message can be:
                    - “Email not valid.”
                    - “Email not present into the database.”
                    - "new_password not valid."
        - Error Response:
            - Type of response: JSON
              (When server can’t process the request, use status code 4xx)
            2. Data in response:{
               error : <message>
               }
                - error message can be:
                    - "Incorrect request."

- **Send Invitation Member Message:**
    - **BACK-END:**
        - URL: “/api/users/invite”
        - Method: POST
        - Type of parameter: JSON
        - Parameter: {email:<string>}
            - email has got correct syntax of a email address.
    - **FRONT-END**
        - Success Response:
            - Type of response: JSON
              (When server process the request, use status code 200)
            1. Data in response: {
               success: <message>
               }
                - success message can be:
                    - “Invitation successfully sent.”
            2. Data in response:{
               error : <message>
               }
                - error message can be:
                    - “User not logged in.”
                    - “User not present into the database.”
                    - “Email not valid.”
        - Error Response:
            - Type of response: JSON
              (When server can’t process the request, use status code 4xx)
            1. Data in response:{
               error : <message>
               }
                - error message can be:
                    - "Incorrect request."

- **Update User Message:**
  - **BACK-END:**
    - URL: “/api/users/update”
    - Method: POST
    - Type of parameter: JSON
    - Parameter: {name:<string>, surname:<string>, role:<string>, photo:<string>}
      - email has got correct syntax of a email address.
      - name is not null.
      - surname is not null.
      - photo is a file.
  - **FRONT-END**
    - Success Response:
      - Type of response: JSON
        (When server process the request, use status code 200)
      1. Data in response: {
         success: <message>
         }
         - success message can be:
           - “User information for " + user_email + " has been updated.”
      2. Data in response:{
         error : <message>
         }
         - error message can be:
           - “User not logged in.”
           - "Mail not present into the database."
           - "Name or surname not valid."
           - "Role or photo not valid."
    - Error Response:
      - Type of response: JSON
        (When server can’t process the request, use status code 4xx)
      1. Data in response:{
         error : <message>
         }
         - error message can be:
           - "Incorrect request."

Projects

- **Add Project Message:**
    - **BACK-END:**
        - URL: “api/projects”
        - Method: POST
        - Type of parameter: JSON
        - Parameter: {project_name:<string>}
            - project_name is not null.
    - **FRONT-END**
        - Success Response:
            - Type of response: JSON
              (When server process the request, use status code 200)
            1. Data in response: {
               success: <message>,
               project_name:<string>,
               points:<string>
               }
                - success message can be:
                    - “New project <project_name_id> added successfully”
            2. Data in response:{
               error : <message>
               }
                - error message can be:
                    - “Project name empty”
                    - “Project name already existing”
                    - “User is not logged in”
                    - “Project name is too long”
        - Error Response:
            - Type of response: JSON
              (When server can’t process the request, use status code 4xx)
            1. Data in response:{
               error : <message>
               }
                - error message can be:
                    - "Incorrect request."

- **Get All Projects of a User:**
  - **BACK-END:**
    - URL: “/api/users/getprojects”
    - Method: POST
    - Type of parameter: JSON
    - Parameter: none.
  - **FRONT-END**
    - Success Response:
      - Type of response: JSON
        (When server process the request, use status code 200)
      1. Data in response: {
         success: <message>,
         projects: [{
                 project_name<string>,
                 points:<string>,
                 owner:<string>
         }, etc]
         - success message can be:
           - “Projects of user.”
         - owner only can be:
           - “true”
           - “false”
      2. Data in response:{
         error : <message>
         }
         - error message can be:
           - “user_email_id is not valid.”
           - “user mail empty
    - Error Response:
      - Type of response: JSON
        (When server can’t process the request, use status code 4xx)
      1. Data in response:{
         error : <message>
         }
         - error message can be:
           - "Incorrect request."

- **Get All Members of a Project Message:**
  - **BACK-END:**
    - URL: “/api/projects/getmembers”
    - Method: POST
    - Type of parameter: JSON
    - Parameter: {project_name_id:<string>}
      - id_project is not null.
  - **FRONT-END**
    - Success Response:
      - Type of response: JSON
        (When server process the request, use status code 200)
      1. Data in response: {
         success: <message>,
         members: [{
         email:<string>,
         name:<string>,
         surname:<string>
         points:<string>
         owner:<string>
         } etc]
         - success message can be:
           - “Members of project " +project_name_id.”
         - owner only can be:
           - “0”
           - “1”
      2. Data in response:{
         error : <message>
         }
         - error message can be:
           - “id_project not valid.”
           - project name empty
    - Error Response:
      - Type of response: JSON
        (When server can’t process the request, use status code 4xx)
      1. Data in response:{
         error : <message>
         }
         - error message can be:
           - "Incorrect request."

- **Rename Project Message:**
    - **BACK-END:**
        - URL: “api/projects/update”
        - Method: POST
        - Type of parameter: JSON
        - Parameter: {old_project_name:<string>, new_project_name:<string>}
            - old_project_name is not null.
            - new_project_name is not null.
    - **FRONT-END**
        - Success Response:
            - Type of response: JSON
              (When server process the request, use status code 200)
            1. Data in response: {
               success: <message>
               }
                - success message can be:
                    - “Project name changed successfully”
            2. Data in response:{
               error : <message>
               }
                - error message can be:
                    - “Old Project name empty”
                    - “New project name empty”
                    - “New project name already existing”
                    - “User is not logged in”
                    - “The user is not an owner”
                    - “New project name too long”
        - Error Response:
            - Type of response: JSON
              (When server can’t process the request, use status code 4xx)
            1. Data in response:{
               error : <message>
               }
                - error message can be:
                    - "Incorrect request."

- **Remove Project Message:**
  - **BACK-END:**
    - URL: “api/projects/remove”
    - Method: POST
    - Type of parameter: JSON
    - Parameter: {project_name:<string>}
      - project_name is not null.
  - **FRONT-END**
    - Success Response:
      - Type of response: JSON
        (When server process the request, use status code 200)
      1. Data in response: {
         success: <message>
         }
         - success message can be:
           - “project <name> removed successfully”
      2. Data in response:{
         error : <message>
         }
         - error message can be:
           - “project_name is empty”
           - “project_name doesn’t exist”
           - “User is not logged in”
           - “User doesn’t own the project”
           - “Project still has some members”
    - Error Response:
      - Type of response: JSON
        (When server can’t process the request, use status code 4xx)
      1. Data in response:{
         error : <message>
         }
         - error message can be:
           - "Incorrect request."

Iterations

- **Add Iteration Message:**
  - **BACK-END:**
    - URL: “api/projects/iterations”
    - Method: POST
    - Type of parameter: JSON
    - Parameter: {project_name:<string>,}
      - project_name is not null.
      - title is not null.
  - **FRONT-END**
    - Success Response:
      - Type of response: JSON
        (When server process the request, use status code 200)
      1. Data in response: {
         success: <message>,
         iteration_number:<string>,
         title:<string>,

         }
         - success message can be:
           - “"Iteration " + l_iteration_data.number.out + " for project " + l_project + " sucessfully created."”
      2. Data in response:{
         error : <message>
         }
         - error message can be:
           - "Project name not valid."
           - “User is not logged in”
           - Project not present into the database.
    - Error Response:
      - Type of response: JSON
        (When server can’t process the request, use status code 4xx)
      1. Data in response:{
         error : <message>
         }
         - error message can be:
           - "Incorrect request."

- **Get All Iterations of a Project Message:**
  - **BACK-END:**
    - URL: “/api/projects/iterations/getprojectiterations”
    - Method: POST
    - Type of parameter: JSON
    - Parameter: {project_name:<string>}
      - project_name is not null.
  - **FRONT-END**
    - Success Response:
      - Type of response: JSON
        (When server process the request, use status code 200)
      1. Data in response: {
         success: <message>,
         iterations: [{iteration_number:<string>,
                 title:<string>,
                 points:<string>}]
         }
         - success message can be:
           - “Iterations of the project " + project_name.”
      2. Data in response:{
         error : <message>
         }
         - error message can be:
           - “project_name not valid.”
    - Error Response:
      - Type of response: JSON
        (When server can’t process the request, use status code 4xx)
      1. Data in response:{
         error : <message>
         }
         - error message can be:
           - "Incorrect request."

- **Delete Iteration of a Project Message:**
  - **BACK-END:**
    - URL: “api/projects/iterations/delete”
    - Method: POST
    - Type of parameter: JSON
    - Parameter: {project_name:<string>, iteration_number:<string>}
      - iteration_number is not null.
      - project_name is not null.
  - **FRONT-END**
    - Success Response:
      - Type of response: JSON
        (When server process the request, use status code 200)
      1. Data in response: {
         success: <message>
         }
         - success message can be:
           - “iteration <name> removed from project <name> successfully”
      2. Data in response:{
         error : <message>
         }
         - error message can be:
           - “User is not logged in”
           - "Project name not valid."
           - "Iteration number not valid."
           - "Cannot delete a BACKLOG iteration."
           - "Project not present into the database."
           - "Iteration not present into the database."
           - "Iteration is not empty."
           - “id_iteration is empty”
           - “id_iteration is doesn’t exist”
      - Type of response: JSON
        (When server can’t process the request, use status code 4xx)
      1. Data in response:{
         error : <message>
         }
         - error message can be:
           - "Incorrect request."

Work Items

- **Add Work Item Message:**
  - **BACK-END:**
    - URL: “api/projects/iterations/workitems”
    - Method: POST
    - Type of parameter: JSON
    - Parameter: {
      project_name_id:<string>,
      iteration_number:<string>,
      work_item_title<string>,
      points:<string>,
      description;<string>,
      comments:[<string>],
      status:<string>,
      links:[<string>]
      }
      - project_name_id is not null.
      - iteration_number is not null.
      - work_item_title is not null.
      - description is not null.
      - comments is a array of comment (strings)
      - status in not null.
      - links is a array of id of work items (strings)
  - **FRONT-END**
    - Success Response:
      - Type of response: JSON
        (When server process the request, use status code 200)
      1. Data in response: {
         success: <message>,
         work_item_id:<string>,
         title:<string>,
         points:<string>
         }
         - success message can be:
           - “New work item <work_item_title> added successfully”
      2. Data in response:{
         error : <message>
         }
         - error message can be:
           - “User is not logged in”
    - Error Response:
      - Type of response: JSON
        (When server can’t process the request, use status code 4xx)
      1. Data in response:{
         error : <message>
         }
         - error message can be:
           - "Incorrect request."

- **Get All Work Items of a Iterations Message:**
    - **BACK-END:**
        - URL: “api/projects/iterations/getworkitems”
        - Method: POST
        - Type of parameter: JSON
        - Parameter: {iteration_number:<string>,project_name:<string>}
            - iteration_number is not null
            - project_name_id is not null.
    - **FRONT-END**
        - Success Response:
            - Type of response: JSON
              (When server process the request, use status code 200)
            1. Data in response: [{
               "id": <string>,
               "number": <string>,
               "nb_iteration": <string>,
               "project": <string>,
               "name": <string>,
               "description": <string>,
               "points": <string>,
               "status": <string>,
               "created_by": <string>,
               "owner": <string>,
               {"sucess": <string>}
               ]
                - success message can be:
                    - “ The work_items of iteration 'nb_iteration' into project 'project_name' are listed above.”
            2. Data in response:{
               error : <message>
               }
                - error message can be:
                    - “iteration_number not valid.”
                    - The author comments 'name_author' doesn't exist into the db.
                    - The work_item name 'name' already exists into project: 'project' and iteraton:’iteration'
                    - The work_item name is empty.
                    - The work_item description is empty.
                    - The work_item description has more then 166536 characters.
                    - The work_item points aren't in the valid range [0-100]
                    - The given iteration doesn't exist.
                    - The given status isn't among the possible choices.
                    - System can't be the owner.
                    - System can't be the creator.
                    - The given user, as regards to the owner, field doesn't exist into the db.
                    - The given user, as regards to the creator, field doesn't exist into the db
        - Error Response:
            - Type of response: JSON
              (When server can’t process the request, use status code 4xx)
            1. Data in response:{
               error : <message>
               }
                - error message can be:
                    - "Incorrect request.

- **Add Link to Work Item Message:**
  - **BACK-END:**
    - URL: “api/projects/iterations/workitems/links”
    - Method: POST
    - Type of parameter: JSON
    - Parameter: {
      id_work_item_source:<string>,
      id_work_item_destination:<string>
      }
      - id_work_item_source is not null.
      - id_work_item_destination is not null.
  - **FRONT-END**
    - Success Response:
      - Type of response: JSON
        (When server process the request, use status code 200)
      1. Data in response: {
         success: <message>,
         }
         - success message can be:
           - “Work item <id_work_item_source> and work item <id_work_item_destination> linked successfully”
      2. Data in response:{
         error : <message>
         }
         - error message can be:
           - “User is not logged in”
           - “Invalid id_work_item_source”
           - “Invalid id_work_item_destination”
           - “Both id_work_item_source and id_work_item_destination are invalid.”
           - “The link (id_work_item_source, id_work_item_destination) is already present into the db.
    - Error Response:
      - Type of response: JSON
        (When server can’t process the request, use status code 4xx)
      1. Data in response:{
         error : <message>
         }
         - error message can be:
           - "Incorrect request."

- **Add Comment to Work Item Message:**
  - **BACK-END:**
    - URL: “api/projects/iterations/workitems/comments”
    - Method: POST
    - Type of parameter: JSON
    - Parameter: {
      work_item_id<string>,
      comment:<string>,
      }
      - work_item_id is not null.
      - comment is not null.
  - **FRONT-END**
    - Success Response:
      - Type of response: JSON
        (When server process the request, use status code 200)
      1. Data in response: {
         success: <message>,
         }
         - success message can be:
           - “New comment added at work item <work_item_title> successfully”
      2. Data in response:{
         error : <message>
         }
         - error message can be:
           - “User is not logged in”
           - “The work_item doesn't exist into the db”
           - “The comment hasn't text.”
           - “System can't be an author."
           - “The user doesn't exist into the db
    - Error Response:
      - Type of response: JSON
        (When server can’t process the request, use status code 4xx)
      1. Data in response:{
         error : <message>
         }
         - error message can be:
           - "Incorrect request."

- **Get Links of Work Items Message:**
    - **BACK-END:**
        - URL: “api/projects/iterations/workitems/getlinks”
        - Method: POST
        - Type of parameter: JSON
        - Parameter: {
          work_item_id<string>
          }
            - work_item_id is not null.
    - **FRONT-END**
        - Success Response:
            - Type of response: JSON
              (When server process the request, use status code 200)
            1. Data in response: {
               success: <message>,
               links:[work_item_id:<string>]
               }
                - success message can be:
                    - “Links of work item <work_item_title>”
            2. Data in response:{
               error : <message>
               }
                - error message can be:
                    - “User is not logged in”
                    - “The given work_item doesn’t exist into the db.
        - Error Response:
            - Type of response: JSON
              (When server can’t process the request, use status code 4xx)
            1. Data in response:{
               error : <message>
               }
                - error message can be:
                    - "Incorrect request."

- **Get Comments of Work Items Message:**
  - **BACK-END:**
    - URL: “api/projects/iterations/workitems/getcomments”
    - Method: POST
    - Type of parameter: JSON
    - Parameter: {
      work_item_id<string>
      }
      - work_item_id is not null.
  - **FRONT-END**
    - Success Response:
      - Type of response: JSON
        (When server process the request, use status code 200)
      1. Data in response: {
         success: <message>,
         comments:[
         date : <string>,
         author: <string>,
         content: <string>
         ]
         }
         - success message can be:
           - “Comments of work item <work_item_title>”
      2. Data in response:{
         error : <message>
         }
         - error message can be:
           - “User is not logged in”
           - “The given work_item doesn’t exist.
    - Error Response:
      - Type of response: JSON
        (When server can’t process the request, use status code 4xx)
      1. Data in response:{
         error : <message>
         }
         - error message can be:
           - "Incorrect request."

- **Delete Work Item Message:**
  - **BACK-END:**
    - URL: “api/projects/iterations/workitems/delete_work_item”
    - Method: POST
    - Type of parameter: JSON
    - Parameter: {
      work_item_id<string>
      }
      - work_item_id is not null.
  - **FRONT-END**
    - Success Response:
      - Type of response: JSON
        (When server process the request, use status code 200)
      - Data in response: {
        success: <message>,
        }
        - success message can be:
          - “Work item <work_item_title> has been deleted”
      2. Data in response:{
         error : <message>
         }
         - error message can be:
           - “User is not logged in”
           - The given work_item doesn’t exist into the db.
    - Error Response:
      - Type of response: JSON
        (When server can’t process the request, use status code 4xx)
      1. Data in response:{
         error : <message>
         }
         - error message can be:
           - "Incorrect request."

- **Update Work Item Message:**
  - **BACK-END:**
    - URL: “api/projects/iterations/workitems/update”
    - Method: POST
    - Type of parameter: JSON
    - Parameter: {
      work_item_id:<string>,
      work_item_title<string>,
      points:<string>,
      description;<string>,
      status:<string>,
      ownedby:<string>,
      projec_namet:<string>
      }
      - work_item_id is not null.
      - work_item_title is not null.
      - points is not null, is a number greater or equals to 0 and less or equals to 100.
      - description is not null.
      - status in not null.
      - ownedby is not null (email of member)
  - **FRONT-END**
    - Success Response:
      - Type of response: JSON
        (When server process the request, use status code 200)
      1. Data in response: {
         success: <message>
         }
         - success message can be:
           - “Work item <work_item_title> has been modified successfully”
      2. Data in response:{
         error : <message>
         }
         - error message can be:
           - “User is not logged in”
           - “New work item name exist”
    - Error Response:
      - Type of response: JSON
        (When server can’t process the request, use status code 4xx)
      1. Data in response:{
         error : <message>
         }
         - error message can be:
           - "Incorrect request."

- Get Work Item Info Message:
  - **BACK-END:**
    - URL: “/api/projects/iterations/getwork_item”
    - Method: POST
    - Type of parameter: JSON
    - Parameter: {
      work_item_id<string>
      }
      - work_item_id is not null.
  - **FRONT-END**
    - Success Response:
      - Type of response: JSON
        (When server process the request, use status code 200)
      - Data in response: {
        id: <string>,
        number: <string>,
        nb_iteration: <string>,
        project: <string>,
        name: <string>,
        description: <string>,
        points: <string>,
        status: <string>,
        created_by: <string>,
        owner: <string>,
        success: <message>
        }
        - success message can be:
          - SUCCESS: The fields of the work_item 'id' are listed above.
      3. Data in response:{
        error : <message>
        }
        - error message can be:
          - The given work_item doesn’t exist into the db.
    - Error Response:
      - Type of response: JSON
        (When server can’t process the request, use status code 4xx)
      2. Data in response:{
        error : <message>
        }
        - error message can be:
          - "Incorrect request."

Search

- **Global Other User Search Message:**
  - **BACK-END:**
    - URL: “api/search/users”
    - Method: POST
    - Type of parameter: JSON
    - Parameter: {
      keyword::<string>
      }
      - keyword is not null.
  - **FRONT-END**
    - Success Response:
      - Type of response: JSON
        (When server process the request, use status code 200)
      1. Data in response: {
         success: <message>,
         users:[email:<string>]
         }
         - success message can be:
           - “Users have been found”
           - “No user has been found”
      2. Data in response:{
         error : <message>
         }
         - error message can be:
           - “User is not logged in”
           - "Email not present into the database."
           - "Keyword not valid."
    - Error Response:
      - Type of response: JSON
        (When server can’t process the request, use status code 4xx)
      1. Data in response:{
         error : <message>
         }
         - error message can be:
           - "Incorrect request."

- **Global Work Item Search Message:**
  - **BACK-END:**
    - URL: “api/search/workitems”
    - Method: POST
    - Type of parameter: JSON
    - Parameter: {
      keyword::<string>
      }
      - keyword is not null.
  - **FRONT-END**
    - Success Response:
      - Type of response: JSON
        (When server process the request, use status code 200)
      1. Data in response: {
         success: <message>,
         users:[work_item_id:<string>]
         }
         - success message can be:
           - “Work Items have been found”
           - “No Work Item has been found”
      2. Data in response:{
         error : <message>
         }
         - error message can be:
           - “User is not logged in”
           - "Email not present into the database."
           - "Keyword not valid."
    - Error Response:
      - Type of response: JSON
        (When server can’t process the request, use status code 4xx)
      1. Data in response:{
         error : <message>
         }
         - error message can be:
           - "Incorrect request."

Members

- **Add Member Message**
  - **BACK-END:**
    - URL: “/api/projects/members/add”
    - Method: POST
    - Type of parameter: JSON
    - Parameter: {project_name_id:<string>,user_email_id:<string>}
      - project_name_id is not null.
      - email has got correct syntax of a email address.
  - **FRONT-END**
    - Success Response:
      - Type of response: JSON
        (When server process the request, use status code 200)
      1. Data in response: {
         success: <message>,
         name:<string>,
         surname:<string>,
         email:<string>,
         points:<string>
         owner:<string>
         }
         - success message can be:
           - “New member <member> added successfully to <project>”
         - owner only can be:
           - “1”
           - “0”
      2. Data in response:{
         error : <message>
         }
         - error message can be:
           - “Email not valid.”
           - "Mail not present into the database."
           - “Project name empty”
           - “New member email empty”
           - “Project name does not exist”
           - “User is not logged in”
           - “The user is not a member of this project”
    - Error Response:
      - Type of response: JSON
        (When server can’t process the request, use status code 4xx)
      2. Data in response:{
         error : <message>
         }
         - error message can be:
           - "Incorrect request."

- **Delete Member Message:**
  - **BACK-END:**
    - URL: “api/projects/members/remove”
    - Method: POST
    - Type of parameter: JSON
    - Parameter: {project_name_id:<string>,user_email_id:<string>}
      - project_name_id is not null.
      - email has got correct syntax of a email address.
  - **FRONT-END**
    - Success Response:
      - Type of response: JSON
        (When server process the request, use status code 200)
      1. Data in response: {
         success: <message>
         }
         - success message can be:
           - “Member <member> removed successfully from <project>”
      2. Data in response:{
         error : <message>
         }
         - error message can be:
           - “Project name empty”
           - “Member email empty”
           - “Project name does not exist”
           - “User is not logged in”
           - “Project name is too long”
           - “The user is not a member of this project”
    - Error Response:
      - Type of response: JSON
        (When server can’t process the request, use status code 4xx)
      3. Data in response:{
         error : <message>
         }
         - error message can be:
           - "Incorrect request."

- **Promote Owner Member Message:**
  - **BACK-END:**
    - URL: “api/projects/members/promote”
    - Method: POST
    - Type of parameter: JSON
    - Parameter: {project_name_id:<string>,user_email_id:<string>}
      - project_name_id is not null.
      - email has got correct syntax of a email address.
  - **FRONT-END**
    - Success Response:
      - Type of response: JSON
        (When server process the request, use status code 200)
      1. Data in response: {
         success: <message>
         }
         - success message can be:
           - “Ne wowner <owner> added successfully to <project>”
      2. Data in response:{
         error : <message>
         }
         - error message can be:
           - “Project name empty”
           - “New owner email empty”
           - “Project name does not exist”
           - “User is not logged in”
           - “Project name too long”
           - “The user is not owner of this project”
    - Error Response:
      - Type of response: JSON
        (When server can’t process the request, use status code 4xx)
      1. Data in response:{
         error : <message>
         }
         - error message can be:
           - "Incorrect request."

- **Owner Member Downgrade Message:**
  - **BACK-END:**
    - URL: “api/projects/members/downgrade”
    - Method: POST
    - Type of parameter: JSON
    - Parameter: {project_name_id:<string>,user_email_id:<string>}
      - project_name_id is not null.
      - email has got correct syntax of a email address.
  - **FRONT-END**
    - Success Response:
      - Type of response: JSON
        (When server process the request, use status code 200)
      1. Data in response: {
         success: <message>
         }
         - success message can be:
           - “Downgrade owner <owner> successfully to <project>”
      2. Data in response:{
         error : <message>
         }
         - error message can be:
           - “Project name empty”
           - “owner email empty”
           - “Project name does not exist”
           - “User is not logged in”
           - “Project name too long”
           - “The user is not owner of this project”
    - Error Response:
      - Type of response: JSON
        (When server can’t process the request, use status code 4xx)
      1. Data in response:{
         error : <message>
         }
         - error message can be:
           - "Incorrect request."