Exercise WebAPI:

Imagine that you are working on a bug tracking project. You have three entities: Project, Bug, User.

The model is described as follows:

User:

* Id - user identifier
* Name - user first name
* Surname - user surname

Bug:

* Id - bug identifier
* projectId - project identifier
* User - related user
* Description - text summary of the bug (maximum 100 characters)
* CreationDate

Project:

* Id - project identifier
* Name - project name
* Description - a brief description of the project (optional)

Additionally, take into account that a Project has Bugs and Bug is assigned to one User.

**Create a WebApi Project from scratch to manage the bug assignment.**

**TASKS**

* Configure the system to use an SQL Server database called BugsManager. Initialize database with 5 users and 5 projects.
* Implement the BugController to Manage Bugs related operations (CRUD) with the following assumptions:

1. Implement an endpoint to assign a bug to a user. This endpoint should be routed as POST /bug and it will receive the following payload

{

“user”: integer (required),

“project”: integer (required),

“description”: string (required)

}

1. Implement an endpoint to return all the bugs for a given user, project, date range or all parameters.

It will handle a GET request of the following form:

/bugs?project\_id=<project-id>&user\_id=<user-id> &start\_date=<start\_date>&end\_date=<end\_date>, where:

* at least one parameter is required;
* only the GET method is allowed, otherwise, a 405 status code is returned;
* if no bugs were found for filter conditions, a 404 status code is returned;
* Otherwise, you should return a 200 status code and response in the following JSON format:

{

"bugs": [

{

"id": <bug-id>,

"description": <bug-description>,

"username": <assigned-username>,

"project": <related-project>,

“CreationDate”: <creationDate>

}

]

}

* Implement, with .net, in a different project, proper screens to support bugs assignment according to implemented web api methods.
  + Dashboard of all bugs. With filtering by user, project and creation date range
  + Create new bug for a project and assign to an existing user.

# Examples

Calling POST /bugs?project\_id=1 will return a 405 status code.

Calling GET /bugs?project\_id=9999 (assuming we do not have a project with id = 9999) will return a 404 status code.

Calling GET /bugs?user\_id=9999 (assuming we do not have a user with id = 9999) will return a404 status code.

Calling GET /bugs?project\_id=1 (assuming we have a project with id = 1) will return a 200 status code and the following response:

{

"bugs": [

{

"id": 1,

"description": "bug 1 description",

"username": "username 1",

"project": "project 1"

},

{

"id": 3,

"description": "bug 3 description",

"username": "username 2",

"project": "project 1"

}

]

}

Calling GET /bugs?project\_id=1&user\_id=2 (assuming we have a project with id = 1 and a user with id = 2) will return a 200 status code and the following response:

{

"bugs": [

{

"id": 3,

"description": "bug 3 description",

"username": "username 2",

"project": "project 1",

“CreationDate”: “dd/MM/yyyy hh:mm:ss”

}

]

}