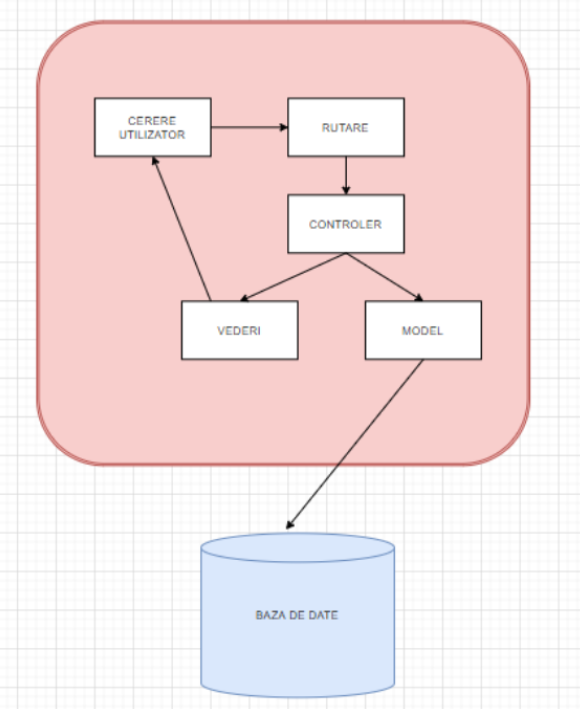
**Data and Domain model**

The main entities I managed to identify are User, Room and Programmes . The relationships between them are from User to Programmes is One-To-Many Relation, from Room to Programmes is also a One-To-Many Relation. I mapped them in the database using migrations. With migrations I created all the fields for every table.

**Application architecture**

****

In my architecture I put also the „Vederi”(Views) in case that this application will also have a frontend part . For the routes part I used the file api.php, there are all my routes that I used in my application . For the Controller I used the ProgrammesController where I implemented all my functions that I used in my application . For Models I used Programmes, Room and User . For the Database I used migrations to create my database .

**Implementation**

**Functionalities**

[x] Brew coffee  
[x] Create programme

For creating a programme I made a simple function called store that contains a request . For inputs I will put all the fields from programmes like name, startingdate, endingdate, participants and room . I will put some values for all of these fields and then it will be created a programme .   
[x] Delete programme

For deleting a programme I made a function called destroy that deletes all the programmes based on a id .

I also made the other 2 crud operations on programmes, like showing all the programmes or updating a programme in the functions show and update .

[x] Book a programme

I made a function called check to see if there are rooms with programmes and then the user will book a programme .

**Business rules**

When creating a programme I made a validation for starting Date and ending Date, because the ending Date must be after the starting Date . Another validation is for the name of the programme, it can be only pilates or kangoojumps .

I also made a validation for checking if there are rooms for making a programme and a validation to see if the user is logged in the site to make a programme .

**Environment**

| **Name** | **Choice** |
| --- | --- |
| Operating system (OS) | Windows 10 |
| Database | MySQL Workbench 8 |
| Web server | XAMPP with Apache and MySQL |
| PHP | 7 with Laravel 8 Framework |
| IDE | PhpStorm |

**Testing**

For testing my application I used Postman .

Example for creating a programme I put my url(<http://127.0.0.1:8000/api/programmes>) and the request POST and there I inserted for example this data :

{

        "name":"KangooJumps",

        "startingdate": "2010-02-06 19:30:13",

        "endingdate": "2010-02-06 20:35:13",

        "participants": 82,

        "room": "room1",

        "created\_at": **null**,

        "updated\_at": **null**

}

Example for deleting a programme I put my url with an ID 7 for example ( <http://127.0.0.1:8000/api/programmes/7> ) and then send .

I also used Postman to test the other CRUD operations like show and update a programme .

**Feedback**

In this section, please let us know what is your opinion about this experience and how we can improve it:

1. Have you ever been involved in a similar experience? If so, how was this one different?
2. Do you think this type of selection process is suitable for you?
3. What's your opinion about the complexity of the requirements?
4. What did you enjoy the most?
5. What was the most challenging part of this anti hackathon?
6. Do you think the time limit was suitable for the requirements?
7. Did you find the resources you were sent on your email useful?
8. Is there anything you would like to improve to your current implementation?
9. What would you change regarding this anti hackathon?