**How concurrency is handled**

When two users are playing the same game and both move the robot at the same time, these changes are saved in the database. If users are playing in the simulation, the game simulation is updated through websockets.

**Calling the API**

**Create a game**

/api/game/create

Arguments:

* **num\_dinos:** Number of dinosaurs in the game.
* **dinos\_pos:** (JSON) A list containing the position of each dinosaur.
* **grid\_cols:** Number of columns in the game grid.
* **grid\_rows:** Number of rows in the game grid.
* **robot\_pos:** (JSON) A list containing the position of the robot, e.g [X, Y].

Request sample:

/api/game/create?num\_dinos=2&dinos\_pos=[[0,3],[7,8]]&grid\_cols=10&grid\_rows=10&robot\_pos=[0,0]

Response sample:

{

"dinosaurs": "{\"1\": {\"X\": 0, \"Y\": 3}, \"2\": {\"X\": 7, \"Y\": 8}}",

"grid\_columns": 10,

"grid\_rows": 10,

"id": 5,

"robot\_id": 5,

"status": "success"

}

**Delete a game**

/api/game/delete

Arguments:

* **id:** The game ID.

Request sample:

/api/game/delete?id=1

Response sample:

{

“message": "The Game 1 has been deleted successfully.",

"status": "success"

}

**Get game data**

/api/game

Arguments:

* **id:** The game ID.

Request sample:

/api/game?id=1

Response sample:

{

"dinosaurs": "{\"1\": {\"X\": 0, \"Y\": 3}, \"2\": {\"X\": 7, \"Y\": 8}}",

"grid\_columns": 10,

"grid\_rows": 10,

"id": 1,

"robot\_id": 1,

"status": "success"

}

**Move a robot**

/api/robot/move

Arguments:

* **id:** The robot ID.
* **direction:** Where to move the robot (left, right, behind, front).

Request sample:

/api/robot/move?id=1&direction=left

Response sample:

{

“message": "The robot has been moved successfully.",

"status": "success"

}

**Robot attack**

/api/robot/attack

Arguments:

* **id:** The ID of the robot.

Request sample:

/api/robot/attack?id=1

Response sample:

{

“message": "Dinosaurs around the robot has been removed successfully.",

"status": "success"

}