Newcomer's test task (hiring process)

Introduction

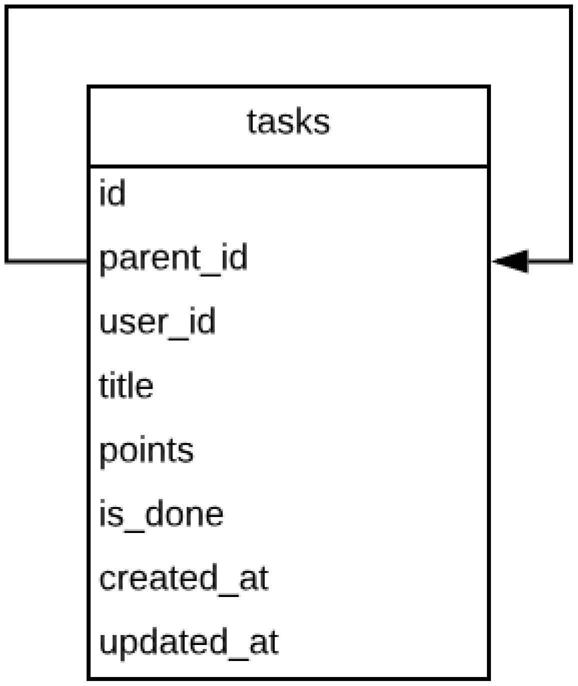
Create tasks management application with simple UI and API. Imagine that application will manage tasks for users that are registered in third party applications. Each task can have subtasks and tasks without subtasks have points. That means the parent task has a sum of points from subtasks. What is more, if all subtasks of a task are done, then the task also is done. If at least one subtask of a task marked as not done then it also becomes not done.

Keep in mind that third party users applications can updates users list at any time.

Requirements

- Create an application using PHP and any framework;
- The application should have simple UI and required API endpoints;
- Create a readme file with instructions on how to run the application;
- If while creating or updating task "is_done" field is set to "true" then and subtasks should be set also as done; If "is_done" field is set to "false" then all parent tasks should be set also as not done;
- If a task has subtasks then task points field value becomes a sum of subtasks points;
- Follow PSR-2 style guide;
- Should be written in OOP;
- Use design patterns when needed;
- Make sure the code in simple and clean;
- Separate concerns;
- Testable code.

Database structure



Note: *a task can have children. Maximum depth: 5.*

Note: *you can have additional fields if needed.*

Required endpoints

Create task endpoint

Method: POST

URI: /api/task

Success response code: 201

Request example:

```
{
   "parent_id":1,
   "user_id":1,
   "title":"Task 1",
   "points":3,
   "is_done":0,
   "email":"john.doe@email.com"
}
```

Response example:

```
{
   "id":1,
   "parent_id":1,
   "user_id":1,
   "title":"Task 1",
   "points":3,
   "is_done":0,
   "created_at":"2020-01-01 00:00:00",
   "updated_at":"2020-01-01 00:00:00"
}
```

Validations:

- parent_id: existing task id or null;
- user_id: required and existing user id;
- title: required;
- point: required integer where the minimum value is 1 and the maximum value is 10;
- is_done: required integer, 0 or 1;

Update task endpoint

Method: PUT

URI: /api/task/{task_id}

Success response code: 201

Request example:

```
{
   "parent_id":1,
   "user_id":1,
   "title":"Task 1",
   "points":10,
   "is_done":1,
   "email":"john.doe@email.com"
}
```

Response example:

```
{
   "id":1,
   "parent_id":1,
   "user_id":1,
   "title":"Task 1",
   "points":10,
   "is_done":1,
   "created_at":"2020-01-01 00:00:00",
   "updated_at":"2020-01-01 00:00:00"
}
```

Validations:

- parent_id: existing task id or null;
- user_id: required and existing user id;
- title: required;
- point: required integer where the minimum value is 1 and the maximum value is 10;
- is_done: required integer, 0 or 1;

Errors handling

- If validations fail should return 400 status code and return validations messages;
- All other unexpected errors should return the 500 status code.

Users management

User should be taken from here: https://gitlab.iterato.lt/snippets/3/raw. Keep in mind that users list endpoint could be changed later.

User interface

Create a basic page where all users with their tasks would be listed. Next to each user should be shown what amount of tasks is done and the total tasks point amount.

John Doe (5/10) • (V) Task 1 (5) • (V) Task 1.1 (2) • (V) Task 1.2 (3) • (X) Task 2 (5) • (X) Task 2.1 (1) • (X) Task 2.2 (2) • (X) Task 2.3 (2)	John Boe (5/12) • (X) Task 1 (12) • (X) Task 1.1 (7) • (V) Task 1.2 (5)	John Toe (14/14) • (V) Task 1 (10) • (V) Task 1.1 (5) • (V) Task 1.2 (5) • (V) Task 1.2.1 (2) • (V) Task 1.2.2 (3) • (V) Task 2 (4) • (V) Task 2.1 (4)