Middle/Senior PHP Developer Position (Laravel)

Project description:

A simple job board where users can post their jobs and send a response to other jobs.

We expect to see the following models:

1. User - user, a default User model, shipped by Laravel framework
2. JobVacancy - a model containing a title and description of the vacancy.
3. JobVacancyResponse - a model for job vacancy responses

Deadlines and workflow:
You can ask any questions related to the test assignment while you are working on it. We judge the work based on how you work with the project, so create a repository for the project and provide a link to the repository before you start. When working on a project, make a commit with a detailed description of the completed task after completing any item.
Also, don't forget to provide a time estimate before starting work and decide which parts of the platform you can implement quickly and which will require significant effort.

Requirements:

Basic system methods:

1. Any user can post a new job vacancy.
2. A user cannot post more than two job vacancies per 24 hours.
3. Any user can send a response to job vacancies posted by other users.
4. Every user can accumulate up to 5 coins.
5. Every user gets new coins every 24 hours. (The amount of cash should be configurable)
6. Users cannot send two or more responses to the same job vacancy.
7. A user can like a job vacancy or another user.
8. To post a job vacancy, a user has to pay two coins and send a response one coin.

Notification system:

You do not need to care about the markup and design—just a technical implementation.

1. Once a new response has been gotten, an email has to be sent to the job vacancy creator.
2. The email has to contain the following details (the job vacancy title, the user's name who sent a response, the total number of responses sent since the job vacancy was posted, and the date when the answer was sent)
3. The system should prevent sending more than one email per hour for the same job vacancy.

Rest API:

Implement rest API endpoints for achieving the following needs:

1. Authenticated users and guests can fetch a list of job vacancies.
2. A list of job vacancies can be sorted by date of creation and by responses count; the list can be filtered by tags and date of creation (day, week, month)
3. Authenticated users and guests can fetch a single job vacancy.
4. Only authenticated users can create a job vacancy.
5. Only owners can update their job vacancies.
6. Only owners can delete their job vacancies (use soft delete)
7. Only authenticated users can send a job vacancy response.
8. Responses can be deleted only by their creators.
9. A user can fetch liked job vacancies, and users

Testing:

Implement feature tests for business methods.

Frontend:

For your convenience, you can use Laravel Nova for the backend. You can use any other libraries at your discretion. We use the following technologies as a stack for the frontend:

- **vue.js** (2 or 3)
- **tailwindcss**

Environment requirements:

- PHP 8

- Compliance with PSR-2 code style. (php_cs.dist.php file included)
- Authorization via Laravel Passport, Laravel Sanctum or JWT
- Upload each milestone to your public GitHub account so that we can see the application's progress.
- Docker ready
- Create readme.md file with the instructions of how to setup the project

php_cs.dist.php

```
<?php

$finder = Symfony\Component\Finder\Finder::create()
    ->in([
        __DIR__.'/app',
        __DIR__.'/database',
        __DIR__.'/routes',
        __DIR__.'/tests',
    ])
    ->name('*.php')
    ->notName('*.blade.php')
    ->ignoreDotFiles(true)
    ->ignoreVCS(true)
;

return (new PhpCsFixer\Config())
    ->setRiskyAllowed(true)
    ->setRules([
        '@PSR2' => true,
        'array_syntax' => ['syntax' => 'short'],
        'ordered_imports' => ['sort_algorithm' => 'alpha'],
        'no_unused_imports' => true,
        'trailing_comma_in_multiline' => true,
        'phpdoc_scalar' => true,
        'unary_operator_spaces' => true,
        'binary_operator_spaces' => true,
        'blank_line_before_statement' => [
            'statements' => ['break', 'continue', 'declare', 'return',
'throw', 'try'],
        ],
        'phpdoc_single_line_var_spacing' => true,
        'phpdoc_var_without_name' => true,
        'class_attributes_separation' => true,
        'method_argument_space' => [
            'on_multiline' => 'ensure_fully_multiline',
            'keep_multiple_spaces_after_comma' => true,
        ],
        'single_trait_insert_per_statement' => true,
        'declare_strict_types' => true,
        'php_unit_method_casing' => ['case' => 'snake_case'],
        'multiline_whitespace_before_semicolons' => ['strategy' =>
'new_line_for_chained_calls'],
    ])
    ->setFinder($finder)
;
```