Pictureworks Laravel Coding Test

Welcome

Hello and thank you for undertaking this PHP test! The project that you would be working on, should you join Pictureworks, is the migration of a non-OO legacy application into the Laravel/Eloquent framework, so we thought the best test to conduct would be to see how you do exactly that! There are no right or wrong answers (provided your solution works), we are simply looking to see your approach to solving the migration problem at hand. Please make sure you keep track of the time taken so that we know how long your specific solution took end-to-end. Thank you again and good luck!

Project Requirement

The zip file provided for you contains a demo of a very simple legacy application. The requirement is to completely port ALL logic, behavior and workflows (see below) included in it to a fresh Laravel application with working data persistency with a database.

Functionality to be Ported

1. GET requests with URL parameters "?id=X" should return the existing styled HTML for some user with id "X". All input must be validated.
2. POST requests with form fields "password", "id" and "comments" will append the value of 'comments' to the existing comments field of user with identifier 'id' provided the 'password' is a given static value. All input must be validated.
3. POST requests with a JSON object containing "password", "id" and "comments" will do exactly the same as (2) above. All input must be validated.
4. Command line executions such as "php controller.php ID COMMENTS" will essentially do the same as (2) above, too, where "ID" is the user identifier and "COMMENTS" is some amount of comments, potentially with spaces. No password is required for this execution type. All input must be validated. Hint: behavior should be ported over to Artisan console commands.
5. Parts 1-4 above should be ported with expected functionality using native Laravel behavior (e.g. url “?id=x” should be available via "/user/{id}").
6. Migrations must be provided. Seeders must be provided. And .env.example file should be filled in with the relevant info.
7. Documentation must list all the routes implemented, params and what they do.

As a general overarching rule: regardless of access method, as long as your code has the exact same input and output as the legacy app, you have done the right thing.

Database Example

```
DROP TABLE IF EXISTS `users`;
CREATE TABLE `users` (
  `id` bigint(20) unsigned NOT NULL AUTO_INCREMENT,
  `name` varchar(255) DEFAULT NULL,
  `comments` text,
  PRIMARY KEY (`id`),
  UNIQUE KEY `id` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8;
INSERT INTO `users` (`id`, `name`, `comments`) VALUES ('1', 'John Smith', 'Directo
r');
```

Deliverables

- A single laravel application that can replace the legacy application provided.

Instructions

1. Create a git repo on GitHub you can choose it to be private or public. Your git commits should begin from the very start (boilerplate) codebase, and should show the natural growth of the solution you provide.
    - Send invite links to GitHub account "pictureworks" as collaborators to your repository.
2. Your repository should show evidence of the following and include the following:
    - Commits showing the growth of the codebase (i.e not one final commit)
    - Installation of a fresh copy of Laravel
    - Migration files
    - Models and relationship definitions (in Eloquent)
    - Use of eloquent models, controllers, routes etc to migrate the behavior of the legacy code provided. How you use, which architecture to follow, and which tools / libraries to use is upto you as long as they are included in your laravel project / composers
    - At least one unit test for one of the functionalities
3. Stack to use.
    - Laravel
    - Postgres
    - Use of the blade templating engine or a mix of solutions you are familiar with for the frontend ****