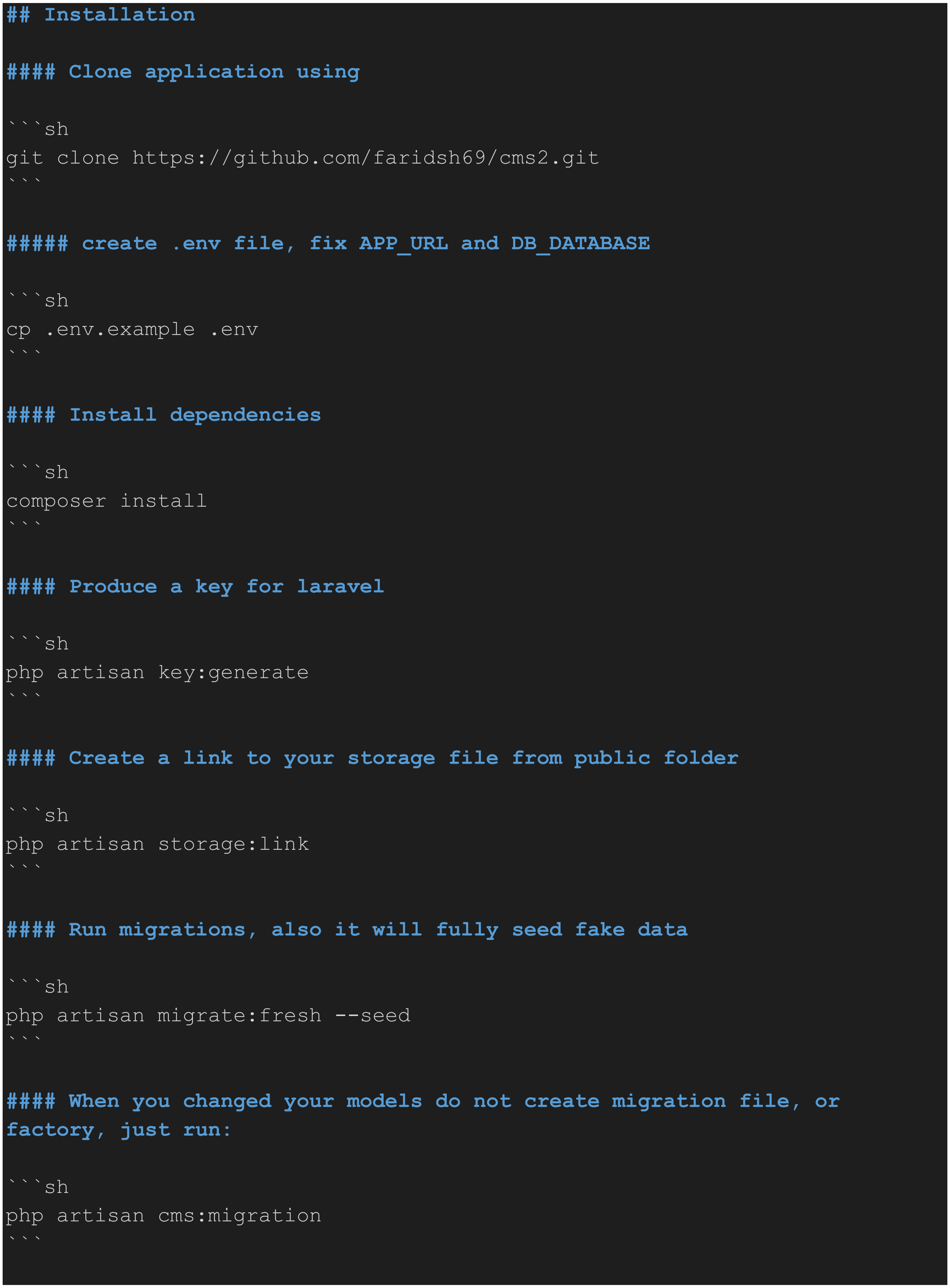
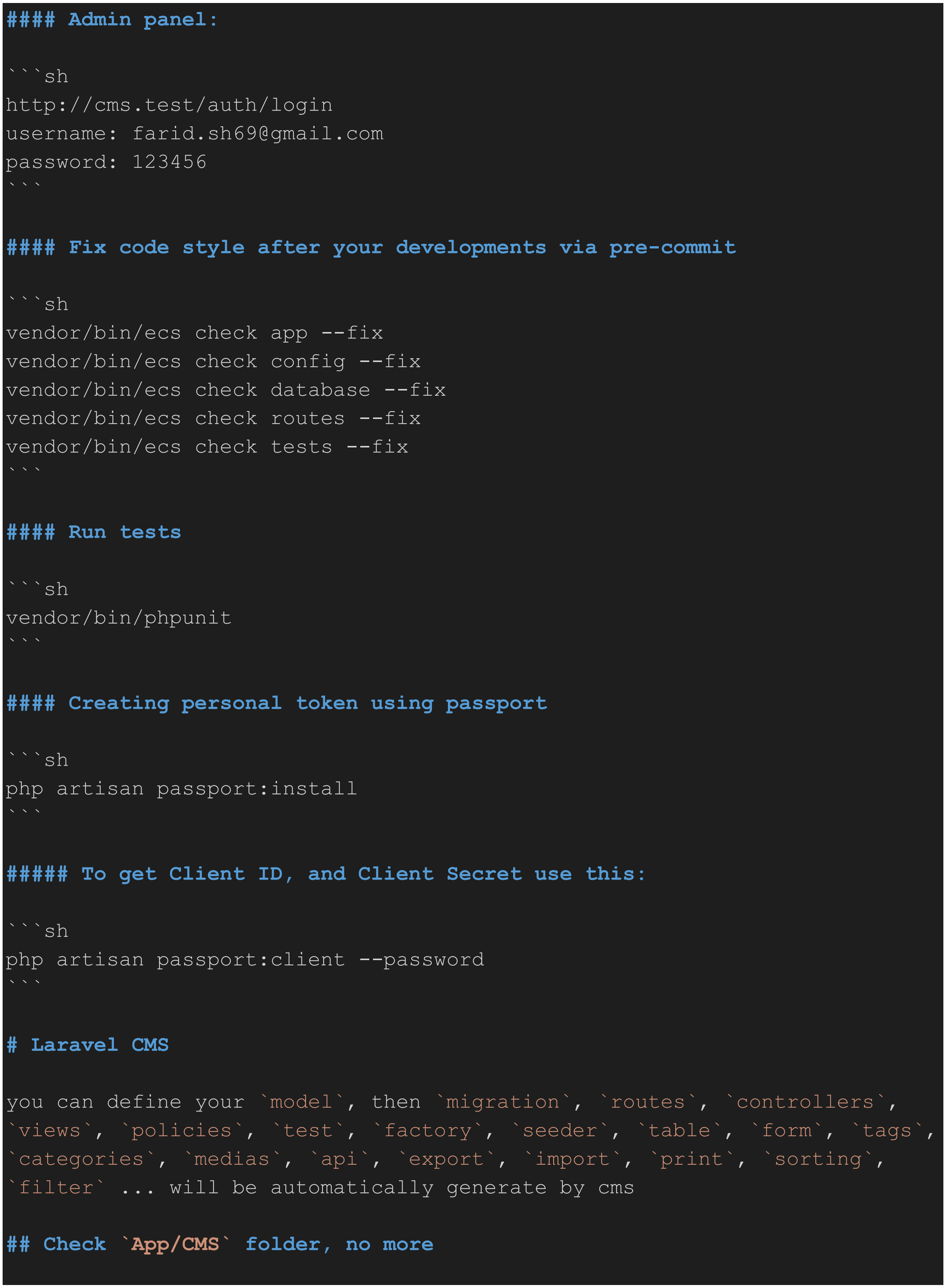


~~~markdown
## Installation

#### Clone application using

```sh
git clone https://github.com/faridsh69/cms2.git
```

##### create .env file, fix APP_URL and DB_DATABASE

```sh
cp .env.example .env
```

#### Install dependencies

```sh
composer install
```

#### Produce a key for laravel

```sh
php artisan key:generate
```

#### Create a link to your storage file from public folder

```sh
php artisan storage:link
```

#### Run migrations, also it will fully seed fake data

```sh
php artisan migrate:fresh --seed
```

#### When you changed your models do not create migration file, or
factory, just run:

```sh
php artisan cms:migration
```
~~~



~~~
#### Admin panel:

```sh
http://cms.test/auth/login
username: farid.sh69@gmail.com
password: 123456
```

#### Fix code style after your developments via pre-commit

```sh
vendor/bin/ecs check app --fix
vendor/bin/ecs check config --fix
vendor/bin/ecs check database --fix
vendor/bin/ecs check routes --fix
vendor/bin/ecs check tests --fix
```

#### Run tests

```sh
vendor/bin/phpunit
```

#### Creating personal token using passport

```sh
php artisan passport:install
```

##### To get Client ID, and Client Secret use this:

```sh
php artisan passport:client --password
```

# Laravel CMS

you can define your `model`, then `migration`, `routes`, `controllers`,
`views`, `policies`, `test`, `factory`, `seeder`, `table`, `form`, `tags`,
`categories`, `medias`, `api`, `export`, `import`, `print`, `sorting`,
`filter` ... will be automatically generate by cms

## Check `App/CMS` folder, no more
~~~

```
Also `Theme` with `modules` and `blocks` with `settings`, `logs`,
`backup`, `code style configuration` are implemented.


### For more information about the structure of model check models folder

<table class="table">
    <thead>
        <th>Attribute</th>
        <th>Use</th>
        <th>Example values</th>
    </thead>
    <tbody>
        <tr>
            <td>Name</td>
            <td>Name of column in database.<br>Name of field in
form.<br>Used in factory.</td>
            <td>title, description, url, content,<br> full_name, image,
language, user_id,<br> category, phone, address, email, password</td>
        </tr>
        <tr>
            <td>Type</td>
            <td>Type of column in database.<br>Used in factory.</td>
            <td><code>string, text, boolean, <br>integer, bigInteger,
tinyInteger, unsignedBigInteger,<br> array, decimal, float, file, date,
time, timestamp</code></td>
        </tr>
        <tr>
            <td>Database</td>
            <td>properties of column in database.</td>
            <td><code>nullable, unsigned, unique, default,
none</code></td>
        </tr>
        <tr>
            <td>Rule</td>
            <td>Rules for store and edit data.</td>
            <td><code>required, nullable, image, max:190,<br> min:3,
unique, boolean, exists:users,id,<br> numeric,
file|image|mimetypes:image/*</code></td>
        </tr>
        <tr>
            <td>Help</td>
            <td>Help block in forms.</td>
            <td>Title should be unique and must not be same with H1.</td>
        </tr>
```

~~~
        <tr>
            <td>Form Type</td>
            <td>Define type of input in form.</td>
            <td><code>textarea, ckeditor, entity, checkbox,<br> enum,
color, date, time, switch,<br> password, email, file, number,
none</code></td>
        </tr>
        <tr>
            <td>Table</td>
            <td>Show field in table or not.</td>
            <td><code>true, false</code></td>
        </tr>
    </tbody>
</table>

```php

class Blog extends App\Services\BaseModel
{
    public $columns = [
        ['name' => 'title'],
        ['name' => 'url'],
        ['name' => 'description'],
        ['name' => 'content'],
        ['name' => 'keywords'],
        ['name' => 'image'],
        ['name' => 'activated'],
        ['name' => 'google_index'],
        ['name' => 'canonical_url']
    ];
}
```
~~~