Особенности создания программных классов

User.php

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Foundation\Auth\User as Authenticatable;

use Illuminate\Notifications\Notifiable;

use Laravel\Passport\HasApiTokens;

class User extends Authenticatable

{

use Notifiable, HasApiTokens, HasFactory;

/\*\*

\* Свойства которые можно заполнить напрямую через функцию fill

\*

\* @var array

\*/

protected $fillable = [

'fullName', 'email', 'birthday', 'pedagogical\_title', 'address', 'phone',

'hiring\_year', 'experience', 'academic\_status\_year',

'scientific\_degree\_year'

];

/\*\*

\* Свойства, которые можно заполнить только через setter/getter

\*

\* @var array

\*/

protected $hidden = [

'password', 'remember\_token',

];

//Отношения с другими моделями

public function department(){

return $this->belongsTo(Department::class);

}

public function commission(){

return $this->belongsTo(Commission::class);

}

public function publications(){

return $this->belongsToMany(Publication::class, 'users\_publications', 'user\_id');

}

public function internships(){

return $this->hasMany(Internship::class);

}

public function qualifications(){

return $this->hasMany(Qualification::class);

}

public function honors(){

return $this->hasMany(Honor::class);

}

public function rebukes(){

return $this->hasMany(Rebuke::class);

}

public function rank(){

return $this->belongsTo(Rank::class);

}

public function educations(){

return $this->hasMany(Education::class);

}

//Помогающие методы

//Преобразовать день рождения к строке

public function getBirthdayString(){

if($this->birthday)

return $this->birthday;

else

return \_\_('messages.notSetted');

}

public function setRole(int $role){

//if this role id in roles array then set it to user otherwise return

if(array\_search($role, \Constants::$roles) !== false)

$this->role = $role;

}

//Convert role to string

public function getRoleString(): string {

return \Constants::$roles[$this->role];

}

//pedagogical title setter

public function setTitle(int $title){

//if exists index in array of titles then change user

if($title >= 0 && $title < sizeof(\Constants::$pedagogicalTitles))

$this->pedagogical\_title = $title;

}

//convert pedagogical title code to string

public function getTitle(): string {

return \Constants::$pedagogicalTitles[$this->pedagogical\_title];

}

//academic status setter

public function setAcademicStatus(int $status){

//if exists index in array then change user

if($status >= 0 && $status < sizeof(\Constants::$academicStatusList))

$this->academic\_status = $status;

}

//convert status code to string

public function getAcademicStatus(): string {

return \Constants::$academicStatusList[$this->academic\_status];

}

public function setScientificDegree(int $scientific){

if($scientific >= 0 && $scientific < sizeof(\Constants::$scientificDegreeList))

$this->scientific\_degree = $scientific;

}

public function getScientificDegree(): string {

return \Constants::$scientificDegreeList[$this->scientific\_degree];

}

public function setDepartment($department){

$this->department\_id = $department;

}

public function getDepartmentID(){

if($this->department)

return $this->department->id;

}

public function getDepartmentName(){

if($this->department)

return $this->department->name;

else

return \_\_('messages.notSetted');

}

public function setCommission($commission){

$this->commission\_id = $commission;

}

public function getCommissionID(){

if($this->commission)

return $this->commission->id;

}

public function getCommissionName(){

if($this->commission)

return $this->commission->name;

else

return \_\_('messages.notSetted');

}

public function setRank($id){

if($id)

$this->rank\_id = $id;

}

public function getRankID(){

if($this->rank)

return $this->rank->id;

}

public function getRankName(){

if(!$this->rank)

return \_\_('messages.notSetted');

return $this->rank->name;

}

public function getShortName(): string {

$fullName = explode(' ', $this->fullName);

//if we have only one word then return without cut

if(sizeof($fullName) == 1){

return $fullName[0];

}

else{

//cut name and return

list($name, $surname) = $fullName;

return $surname . ' ' . mb\_substr($name, 0, 1) . '.';

}

}

public function getAvatar(){

if($this->avatar)

return $this->avatar;

else

return env('APP\_URL') . '/storage/avatars/noAva.jpg';

}

//generate secret values

public function generatePassword($password){

if($password){

$this->password = bcrypt($password);

}

}

public function cryptPassport($passport){

if(!$passport)

return;

$this->passport = encrypt($passport);

$this->save();

}

public function cryptCode($code){

if(!$code)

return;

$this->code = encrypt($code);

$this->save();

}

public function getToken(bool $long = false){

$token = $this->createToken(config('app.name'));

return $token->accessToken;

}

}