Розробки алгоритмів методів програмних класів або процедур/функцій з урахуванням особливостей взаємодії з СУБД

UserRepository.php

class UserRepository extends BaseRepository implements UserRepositoryInterface

{

private $avatarService;

private $model = User::class;

private $sortFields = [

'ID' => 'id',

'name' => 'fullName',

'email' => 'email'

];

public function \_\_construct(PhotoUploader $avatarService)

{

$this->avatarService = $avatarService;

}

public function createRules(array $inputData): array

{

$rules = [];

if($inputData['filterName'] ?? null)

$rules[] = new LikeRule('fullName', $inputData['filterName']);

if($inputData['filterEmail'] ?? null)

$rules[] = new LikeRule('email', $inputData['filterEmail']);

if($inputData['filterCommission'] ?? null)

$rules[] = new EqualRule('commission\_id', $inputData['filterCommission']);

if($inputData['filterDepartment'] ?? null)

$rules[] = new EqualRule('department\_id', $inputData['filterDepartment']);

if($inputData['filterRank'] ?? null)

$rules[] = new EqualRule('rank\_id', $inputData['filterRank']);

if($inputData['filterTitle'] ?? null)

$rules[] = new EqualRule('pedagogical\_title', $inputData['filterTitle']);

if($inputData['filterCategory'] ?? null) {

//$rules[] = new EqualRule('pedagogical\_title', $inputData['filterTitle']);

}

$rules = array\_merge($rules, $this->createSortRules($inputData['sort'] ?? null, $this->sortFields));

return $rules;

}

public function getModel(): Model

{

return app($this->model);

}

public function all()

{

return $this->getModel()->all();

}

public function getForCombo()

{

return $this->getModel()->all('id', 'name', 'surname', 'patronymic');

}

public function getForExportList(): array

{

$users = $this->getModel()->all('id', 'fullName')->toArray();

return to\_export\_list($users);

}

}

RebukeRepository.php

class RebukeRepository extends BaseRepository implements RebukeRepositoryInterface

{

private $model = Rebuke::class;

protected $sortFields = [

'ID' => 'id',

'title' => 'title',

'datePresentation' => 'date\_presentation'

];

public function createRules(array $inputData): array

{

$rules = [];

if($inputData['user\_id'] ?? null)

$rules[] = new EqualRule('user\_id', $inputData['user\_id']);

if($inputData['filterUser'] ?? null)

$rules[] = new EqualRule('user\_id', $inputData['filterUser']);

if($inputData['filterTitle'] ?? null)

$rules[] = new LikeRule('title', $inputData['filterTitle']);

if($inputData['filterFrom'] ?? null)

$rules[] = new DateMoreRule('date\_presentation', $inputData['filterFrom']);

if($inputData['filterTo'] ?? null)

$rules[] = new DateLessRule('date\_presentation', $inputData['filterTo']);

$rules = array\_merge($rules, $this->createSortRules($inputData['sort'] ?? null, $this->sortFields));

return $rules;

}

public function getModel(): Model

{

return app($this->model);

}

public function all()

{

return $this->getModel()->all();

}

public function paginateForUser($user\_id, ?int $size = null)

{

$size = $size ?? config('app.PAGINATE\_SIZE', 10);

return $this->getModel()->query()->where('user\_id', $user\_id)->paginate($size);

}

public function getUserString(int $user\_id): string

{

//get all rebukes

$rebukes = $this->getModel()->query()->where('user\_id', $user\_id)->get();

//parse string

$rebukesString = $rebukes->reduce(function(string $acc, $item){

return $acc . implode(', ', [$item->title, $item->date\_presentation, $item->order]) . ';';

}, '');

//return info

return $rebukesString ? $rebukesString : 'Немає інформації';

}

}

PublicationRepository.php

class PublicationRepository extends BaseRepository implements PublicationRepositoryInterface

{

private $model = Publication::class;

protected $sortFields = [

'ID' => 'id',

'title' => 'title',

'date' => 'date\_of\_publication'

];

public function getModel(): Model

{

return app($this->model);

}

public function createRules(array $inputData): array

{

$rules = [];

if($inputData['user\_id'] ?? null)

$rules[] = new HasAssociateRule('authors',

new EqualRule('users.id', $inputData['user\_id']));

if($inputData['filterTitle'] ?? null)

$rules[] = new LikeRule('title', $inputData['filterTitle']);

if($inputData['filterFrom'] ?? null)

$rules[] = new DateMoreRule('date\_of\_publication', $inputData['filterFrom']);

if($inputData['filterTo'] ?? null)

$rules[] = new DateLessRule('date\_of\_publication', $inputData['filterTo']);

$rules = array\_merge($rules, $this->createSortRules($inputData['sort'] ?? null, $this->sortFields));

return $rules;

}

public function all()

{

return $this->getModel()->all();

}

public function paginateForUser($user\_id, ?int $size = null)

{

$size = $size ?? config('app.PAGINATE\_SIZE', 10);

return $this->getModel()->query()->whereHas('authors', function (Builder $q) use($user\_id){

$q->where('user\_id', $user\_id);

})->paginate($size);

}

}