

Код системы

`.env.example``APP_NAME=Laravel``APP_ENV=local``APP_KEY=``APP_DEBUG=true``APP_URL=http://localhost``LOG_CHANNEL=stack``LOG_DEPRECATIONS_CHANNEL=null``LOG_LEVEL=debug``DB_CONNECTION=mysql``DB_HOST=127.0.0.1``DB_PORT=3306``DB_DATABASE=laravel``DB_USERNAME=root``DB_PASSWORD=``LDAP_HOSTS=``LDAP_BASE_DN=``LDAP_USERNAME=``LDAP_PASSWORD=``LDAP_ACCOUNT_PREFIX=``LDAP_ACCOUNT_SUFFIX=``BROADCAST_DRIVER=log``CACHE_DRIVER=file``FILESYSTEM_DISK=local`

QUEUE\_CONNECTION=sync

SESSION\_DRIVER=file

SESSION\_LIFETIME=120

MEMCACHED\_HOST=127.0.0.1

REDIS\_HOST=127.0.0.1

REDIS\_PASSWORD=null

REDIS\_PORT=6379

MAIL\_MAILER=smtp

MAIL\_HOST=mailhog

MAIL\_PORT=1025

MAIL\_USERNAME=null

MAIL\_PASSWORD=null

MAIL\_ENCRYPTION=null

MAIL\_FROM\_ADDRESS="hello@example.com"

MAIL\_FROM\_NAME="\${APP\_NAME}"

AWS\_ACCESS\_KEY\_ID=

AWS\_SECRET\_ACCESS\_KEY=

AWS\_DEFAULT\_REGION=us-east-1

AWS\_BUCKET=

AWS\_USE\_PATH\_STYLE\_ENDPOINT=false

PUSHER\_APP\_ID=

PUSHER\_APP\_KEY=

PUSHER\_APP\_SECRET=

PUSHER\_APP\_CLUSTER=mt1

MIX\_PUSHER\_APP\_KEY="\${PUSHER\_APP\_KEY}"

```
MIX_PUSHER_APP_CLUSTER="${PUSHER_APP_CLUSTER}"
```

```
app/Console/Commands/AddAPIUser.php
```

```
<?php
```

```
namespace App\Console\Commands;
```

```
use Illuminate\Support\Facades\Hash;
```

```
use App\Models\User;
```

```
use Illuminate\Console\Command;
```

```
class AddAPIUser extends Command
```

```
{
```

```
    /**
```

```
     * The name and signature of the console command.
```

```
     *
```

```
     * @var string
```

```
     */
```

```
    protected $signature = 'user:add {nick} {password}';
```

```
    /**
```

```
     * The console command description.
```

```
     *
```

```
     * @var string
```

```
     */
```

```
    protected $description = 'Add new user for API access';
```

```
    /**
```

```
     * Create a new command instance.
```

```
     *
```

```
     * @return void
```

```

*/
public function __construct()
{
    parent::__construct();
}

/**
 * Execute the console command.
 *
 * @return int
 */
public function handle(User $user)
{
    //$user = new \App\User();
    $user->nick    = $this->argument('nick');
    $user->password = Hash::make($this->argument('password'));

    $user->save();
}
}

```

app/Console/Commands/AddAPIUserToken.php

```
<?php
```

```

namespace App\Console\Commands;
use App\Models\User;
use Illuminate\Console\Command;

class AddAPIUserToken extends Command
{
    /**

```

```
* The name and signature of the console command.
```

```
*
```

```
* @var string
```

```
*/
```

```
protected $signature = 'user:add_token {nick} {token_name}';
```

```
/**
```

```
* The console command description.
```

```
*
```

```
* @var string
```

```
*/
```

```
protected $description = 'Add API token for user';
```

```
/**
```

```
* Create a new command instance.
```

```
*
```

```
* @return void
```

```
*/
```

```
public function __construct()
```

```
{
```

```
    parent::__construct();
```

```
}
```

```
/**
```

```
* Execute the console command.
```

```
*
```

```
* @return int
```

```
*/
```

```
public function handle()
```

```
{
```

```
    $user = User::where('nick', $this->argument('nick'))->first();
```

```

    $newToken = $user->createToken($this->argument('token_name'));
    print $newToken->plainTextToken . PHP_EOL;
    return 0;
}
}

```

app/Exceptions/Handler.php

```
<?php
```

```
namespace App\Exceptions;
```

```
use Illuminate\Foundation\Exceptions\Handler as ExceptionHandler;
```

```
use Throwable;
```

```
class Handler extends ExceptionHandler
```

```
{
```

```
    /**
```

```
     * A list of the exception types that are not reported.
```

```
     *
```

```
     * @var array<int, class-string<Throwable>>
```

```
     */
```

```
    protected $dontReport = [
```

```
        //
```

```
    ];
```

```
    /**
```

```
     * A list of the inputs that are never flashed for validation exceptions.
```

```
     *
```

```
     * @var array<int, string>
```

```
     */
```

```
    protected $dontFlash = [
```

```

        'current_password',
        'password',
        'password_confirmation',
    ];

    /**
     * Register the exception handling callbacks for the application.
     *
     * @return void
     */
    public function register()
    {
        $this->reportable(function (Throwable $e) {
            //
        });
    }
}

app/Http/Actions/ReportHours/ActionShowReportHours.php
<?php

namespace App\Http\Actions\ReportHours;

use App\Http\Requests\ReportHours\GetByStartDateRequest;
use App\Http\Resources\ReportHours\ShortReportHoursCollection;
use App\Models\ReportHours;

class ActionShowReportHours
{
    /**
     * Show report

```

\*

\* Getting a short report on hours from employees for a certain period  
\* of days.

\*

\* @group Report hours

\*/

```
function __invoke(GetByStartDateRequest $request) {
    $report = new ReportHours(
        $request->validated('start_date'),
        $request->validated('end_date')
    );
    return new ShortReportHoursCollection($report->reportShort());
}
}
```

app/Http/Actions/ReportHours/ActionShowReportHoursByProject.php

<?php

```
namespace App\Http\Actions\ReportHours;
```

```
use App\Http\Requests\ReportHours\GetByStartDateRequest;
```

```
use App\Http\Resources\ReportHours\ReportHoursWithProjectCollection;
```

```
use App\Models\ReportHours;
```

```
class ActionShowReportHoursByProject
```

```
{
```

```
/**
```

```
 * Report hours by project
```

```
 *
```

```
 * Getting a report on hours from employees ordered by project
```

```
 * for a certain period of days.
```



```

*
* If several employees worked on one project, then several records
* are created in the `reports` array with the same value of the fields
* related to the project.
*
* @group Report hours
*/
function __invoke(GetByStartDateRequest $request) {
    $report = new ReportHours(
        $request->validated('start_date'),
        $request->validated('end_date')
    );
    return new ReportHoursWithProjectCollection($report-
>reportByProject());
}
}

app/Http/Actions/Schedule/ActionScheduleOfficial.php

<?php
namespace App\Http\Actions\Schedule;

use App\Http\Requests\ReportHours\GetByStartDateRequest;
use App\Operations\Schedule\GetterVacationOfficial;
use App\Operations\Schedule\RangeDate;

class ActionScheduleOfficial
{
    public function __invoke(GetByStartDateRequest $request)
    {
        $range = RangeDate::createFromStrings(
            $request->validated('start_date'),

```

```

        $request->validated('end_date')
    );
    $getter = new GetterVacationOfficial($range);
    $schedule = $getter->schedule;
    return OfficialDayResource::collection($schedule->getDays());
}
}

```

app/Http/Actions/Schedule/ActionWorkersSchedule.php

```

<?php
namespace App\Http\Actions\Schedule;

use App\Http\Requests\ReportHours\GetByStartDateRequest;
use App\Operations\Schedule\RangeDate;
use App\Operations\Schedule\GetterWorkerVacation;

class ActionWorkersSchedule
{
    public function __invoke(GetByStartDateRequest $request)
    {
        $range = RangeDate::createFromStrings(
            $request->validated('start_date'),
            $request->validated('end_date')
        );
        $getter = new GetterWorkerVacation($range);
        return $getter->getScheduleWithWorkersHours();
    }
}

```

app/Http/Actions/Schedule/OfficialDayResource.php

```

<?php

```

```

namespace App\Http\Actions\Schedule;

use Illuminate\Http\Resources\Json\JsonResource;

class OfficialDayResource extends JsonResource
{
    /**
     * Transform the resource into an array.
     *
     * @param \Illuminate\Http\Request $request
     * @return array|\Illuminate\Contracts\Support\Arrayable|\
JsonSerializable
     */
    public function toArray($request)
    {
        return [
            'status' => $this->resource->getStatus(),
            'info' => $this->resource->getInfo(),
        ];
    }
}

```

app/Http/Actions/Token/Create/ActionCreateToken.php

```
<?php
```

```
namespace App\Http\Actions\Token\Create;
```

```
use App\Http\Resources\NewTokenResource;
```

```
class ActionCreateToken
```

```

{
    /**
     * Token generation
     *
     * @group Tokens
     * @response 201 {
     *     "data": {
     *         "name": "report-service",
     *         "token": "{YOUR_AUTH_KEY}"
     *     }
     * }
     */
    function __invoke(CreateRequest $request)
    {
        $newToken = $request->user()->createToken($request->name);
        return (new NewTokenResource($newToken))->code(201);
    }
}

```

app/Http/Actions/Token/Create/CreateRequest.php

<?php

namespace App\Http\Actions\Token\Create;

use App\Http\Requests\BaseRequest;

class CreateRequest extends BaseRequest

```

{
    /**
     * Get the validation rules that apply to the request.
     *

```

```

* @return array
*/
public function rules()
{
    return [
        'name' => [
            'required',
            'unique:Laravel\Sanctum\PersonalAccessToken,name',
        ],
    ];
}

public function bodyParameters()
{
    return [
        'name' => [
            'description' => 'Token name',
            'example' => 'report-service',
        ],
    ];
}
}

app/Http/Actions/Token/Delete/ActionDeleteOneToken.php

<?php

namespace App\Http\Actions\Token\Delete;

use Illuminate\Http\Request;

class ActionDeleteOneToken

```

```

{
    /**
     * Deleting a token
     *
     * Removing an authorized user token by token name.
     *
     * @urlParam token_name string required user token name Example:
report-service
     * @group Tokens
     * @response 204
     */
    function __invoke(Request $request, string $tokenName)
    {
        $request->user()
            ->tokens()
            ->where('name', $tokenName)
            ->firstOrFail()
            ->delete();
        return response("", 204);
    }
}

```

app/Http/Actions/Token/Read/ActionReadAllTokens.php

<?php

namespace App\Http\Actions\Token\Read;

use App\Http\Actions\Token\Resources\TokenCollection;

use Illuminate\Support\Facades\Auth;

use Illuminate\Support\Facades\Request;

```
class ActionReadAllTokens
```

```
{
    /**
     * Read all authorized user tokens
     *
     * @group Tokens
     */
    function __invoke(Request $request)
    {
        return new TokenCollection(Auth::user()->tokens);
    }
}
```

```
app/Http/Actions/Token/Read/ActionReadOneToken.php
```

```
<?php
```

```
namespace App\Http\Actions\Token\Read;
```

```
use App\Http\Actions\Token\Resources\UserTokenResource;
```

```
use Illuminate\Support\Facades\Auth;
```

```
class ActionReadOneToken
```

```
{
    /**
     * Read an authorized user tokens
     *
     * @urlParam token_name string required user token name Example:
```

```
report-service
```

```
    * @group Tokens
```

```
    */
```

```
    function __invoke(string $tokenName)
```

```

{
    return new UserTokenResource(
        Auth::user()
            ->tokens()
            ->where('name', $tokenName)
            ->firstOrFail()
    );
}
}

```

app/Http/Actions/Token/Resources/TokenCollection.php

<?php

```
namespace App\Http\Actions\Token\Resources;
```

```
use Illuminate\Http\Resources\Json\ResourceCollection;
```

```
class TokenCollection extends ResourceCollection
```

```

{
    /**
     * Transform the resource collection into an array.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return array|\Illuminate\Contracts\Support\Arrayable|\
    JsonSerializable
     */
    public function toArray($request)
    {
        return [
            'tokens' => UserTokenResource::collection($this->collection),
        ];
    }
}

```



```

    }
}

app/Http/Actions/Token/Resources/UserTokenResource.php

<?php

namespace App\Http\Actions\Token\Resources;

use Illuminate\Http\Resources\Json\JsonResource;

class UserTokenResource extends JsonResource
{
    /**
     * Transform the resource into an array.
     *
     * @param \Illuminate\Http\Request $request
     * @return array|\Illuminate\Contracts\Support\Arrayable|\
JsonSerializable
     */
    public function toArray($request)
    {
        return [
            'name' => $this->name,
            'abilities' => $this->abilities,
            'last_used_at' => $this->last_used_at,
        ];
    }
}

app/Http/Actions/Token/Update/ActionUpdateOneToken.php

<?php

```

```

namespace App\Http\Actions\Token\Update;

use App\Http\Actions\Token\Resources\UserTokenResource;
use Illuminate\Support\Facades\Auth;

class ActionUpdateOneToken
{
    /**
     * Edit token
     *
     * Edit token name and rights.
     * Rights will be used only those that are available to the user of the
     * token.
     *
     * Token key stays the same.
     *
     * @urlParam token_name string required user token name Example:
report-service
     * @group Tokens
     * @response {
     *   "data": {
     *     "name": "report-service",
     *     "abilities": [
     *       "report:getShort",
     *       "report:getByProject"
     *     ],
     *     "last_used_at": null
     *   }
     * }
    */

```

```

function __invoke(UpdateRequest $request, string $tokenName)
{
    $token = Auth::user()
        ->tokens()
        ->where('name', $tokenName)
        ->firstOrFail();
    if ($request->name)
    {
        $token->name = $request->name;
    }
    if ($request->abilities)
    {
        $token->abilities = $request->abilities;
    }
    $token->save();
    return new UserTokenResource($token);
}
}

```

app/Http/Actions/Token/Update/UpdateRequest.php

```
<?php
```

```
namespace App\Http\Actions\Token\Update;
```

```
use App\Http\Requests\BaseRequest;
```

```
class UpdateRequest extends BaseRequest
```

```

{
    /**
     * Get the validation rules that apply to the request.
     *

```

```

* @return array
*/
public function rules()
{
    return [
        'name' => ['unique:Laravel\Sanctum\PersonalAccessToken,name'],
        'abilities' => ['array'],
        'abilities.*' => ['string'],
    ];
}

public function bodyParameters()
{
    return [
        'name' => [
            'description' => 'Token name',
            'example' => 'report-service',
        ],
        'abilities' => [
            'description' => 'Token rules',
            'example' => ['*'],
        ],
    ];
}
}

```

app/Http/Actions/User/ActionLogin.php

```
<?php
```

```
namespace App\Http\Actions\User;
```

```

use App\Http\Requests\LoginRequest;
use App\Http\Resources\NewTokenResource;
use Illuminate\Support\Facades\Auth;

class ActionLogin
{
    /**
     * Authentication
     *
     * User authentication.
     * A token is issued for the user's nick and password for further
     * interaction with the api.
     *
     * The token is created under the name "login" and is available like other
     * tokens.
     *
     * If a "login" token previously existed, it is removed and a new user
token
     * is generated.
     *
     * @group Users
     * @unauthenticated
     * @response 201 {
     *   "data": {
     *     "name": "login",
     *     "token": "{YOUR_AUTH_KEY}"
     *   }
     * }
     */
    function __invoke(LoginRequest $request)
    {

```

```

        if(!Auth::attempt($request->validated()))
        {
            return response('Unauthorized', 401);
        }
        $request->user()->tokens()->where('name', 'login')->delete();
        $newToken = $request->user()->createToken('login');
        return (new NewTokenResource($newToken));
    }
}

```

app/Http/Middleware/Authenticate.php

```
<?php
```

```
namespace App\Http\Middleware;
```

```
use Illuminate\Auth\Middleware\Authenticate as Middleware;
```

```
class Authenticate extends Middleware
```

```
{
```

```
    /**
```

```
        * Get the path the user should be redirected to when they are not
authenticated.
```

```
        *
```

```
        * @param \Illuminate\Http\Request $request
```

```
        * @return string|null
```

```
        */
```

```
protected function redirectTo($request)
```

```
{
```

```
    if (! $request->expectsJson()) {
```

```
        return route('login');
```

```
    }
```

```

    }
}

app/Http/Middleware/XAuthKeyHeader2Bearer.php

<?php

namespace App\Http\Middleware;

use Closure;
use Illuminate\Http\Request;

class XAuthKeyHeader2Bearer
{
    /**
     * Handle an incoming request.
     *
     * @param \Illuminate\Http\Request $request
     * @param \Closure(\Illuminate\Http\Request): (\Illuminate\Http\
Response|\Illuminate\Http\RedirectResponse) $next
     * @return \Illuminate\Http\Response|\Illuminate\Http\RedirectResponse
     */
    public function handle(Request $request, Closure $next)
    {
        if($request->hasHeader('X-Auth-Key'))
        {
            $request->headers->add([
                'Authorization' => 'Bearer ' . $request->header('X-Auth-Key')
            ]);
        }
        return $next($request);
    }
}

```

```
}
```

```
app/Http/Requests/BaseRequest.php
```

```
<?php
```

```
namespace App\Http\Requests;
```

```
use Illuminate\Foundation\Http\FormRequest;
```

```
abstract class BaseRequest extends FormRequest
```

```
{
```

```
    /**
```

```
        * Авторизовать пользователя следует при помощи UserPolicy и
middleware
```

```
        *
```

```
        * @return bool
```

```
        */
```

```
    public function authorize()
```

```
    {
```

```
        return true;
```

```
    }
```

```
}
```

```
app/Http/Requests/LoginRequest.php
```

```
<?php
```

```
namespace App\Http\Requests;
```

```
class LoginRequest extends BaseRequest
```

```
{
```

```
    /**
```



```
* Get the validation rules that apply to the request.
```

```
*
```

```
* @return array
```

```
*/
```

```
public function rules()
```

```
{
```

```
    return [
```

```
        'nick' => ['required'],
```

```
        'password' => ['required'],
```

```
    ];
```

```
}
```

```
public function bodyParameters()
```

```
{
```

```
    return [
```

```
        'nick' => [
```

```
            'description' => 'User nick',
```

```
            'example' => 'testuser',
```

```
        ],
```

```
        'password' => [
```

```
            'description' => 'User password',
```

```
            'example' => 'password',
```

```
        ],
```

```
    ];
```

```
}
```

```
}
```

```
app/Http/Requests/ReportHours/GetByStartDateRequest.php
```

```
<?php
```

```
namespace App\Http\Requests\ReportHours;
```

```

use App\Http\Requests\BaseRequest;

class GetByStartDateRequest extends BaseRequest
{
    /**
     * Get the validation rules that apply to the request.
     *
     * @return array
     */
    public function rules()
    {
        return [
            'start_date' => ['date', 'nullable'],
            'end_date' => ['date', 'nullable'],
        ];
    }

    public function bodyParameters()
    {
        return [
            'start_date' => [
                'description' => 'Start date of the reporting period. Inclusive.',
                'example' => '2001-10-08',
            ],
            'end_date' => [
                'description' => 'End date of the reporting period. Inclusive.',
                'example' => '2016-10-08',
            ],
        ];
    }
}

```

```
}
```

```
app/Http/Resources/BaseResource.php
```

```
<?php
```

```
namespace App\Http\Resources;
```

```
use Illuminate\Http\Resources\Json\JsonResource;
```

```
abstract class BaseResource extends JsonResource
```

```
{
```

```
    public function code(int $code)
```

```
    {
```

```
        return $this->response()->setStatusCode($code);
```

```
    }
```

```
}
```

```
app/Http/Resources/NewTokenResource.php
```

```
<?php
```

```
namespace App\Http\Resources;
```

```
use Illuminate\Http\Resources\Json\JsonResource;
```

```
class NewTokenResource extends BaseResource
```

```
{
```

```
    /**
```

```
     * Transform the resource into an array.
```

```
     *
```

```
     * @param \Illuminate\Http\Request $request
```

```

        *   @return   array|\Illuminate\Contracts\Support\Arrayable|\
JsonSerializable
    */
    public function toArray($request)
    {
        return [
            'name' => $this->resource->accessToken->name,
            'token' => $this->resource->plainTextToken,
        ];
    }
}

```

app/Http/Resources/ReportHours/ReportHoursWithProjectCollection.php

```
<?php
```

```
namespace App\Http\Resources\ReportHours;
```

```
use Illuminate\Http\Resources\Json\ResourceCollection;
```

```
class ReportHoursWithProjectCollection extends ResourceCollection
```

```

{
    /**
     * Transform the resource collection into an array.
     *
     * @param  \Illuminate\Http\Request  $request
     *   @return   array|\Illuminate\Contracts\Support\Arrayable|\

```

```
JsonSerializable
```

```

    */
    public function toArray($request)
    {
        return [

```

```

        'count_reports' => count($this->collection),
        'reports' => ReportHoursWithProjectResource::collection(
            $this->collection
        ),
    ];
}
}

```

app/Http/Resources/ReportHours/ReportHoursWithProjectResource.php

```
<?php
```

```
namespace App\Http\Resources\ReportHours;
```

```
use Illuminate\Http\Resources\Json\JsonResource;
```

```
class ReportHoursWithProjectResource extends JsonResource
```

```

{
    /**
     * Transform the resource into an array.
     *
     * @param  \Illuminate\Http\Request $request
     * @return array|\Illuminate\Contracts\Support\Arrayable|\

```

JsonSerializable

```
*/
```

```
public function toArray($request)
```

```
{
```

```
    return [
```

```
        'nick' => $this->resource->nick,
```

```
        'name' => $this->resource->name,
```

```
        'company' => $this->resource->company,
```

```
        'customer_id' => $this->resource->customer_id,
```

```

        'customer_name' => $this->resource->customer_name,
        'project_id' => $this->resource->project_id,
        'project_name' => $this->resource->project_name,
        'total_hours' => $this->resource->total_hours,
    ];
}
}

```

app/Http/Resources/ReportHours/ShortReportHoursCollection.php

```
<?php
```

```
namespace App\Http\Resources\ReportHours;
```

```
use Illuminate\Http\Resources\Json\ResourceCollection;
```

```
class ShortReportHoursCollection extends ResourceCollection
```

```

{
    /**
     * Transform the resource collection into an array.
     *
     * @param \Illuminate\Http\Request $request
     * @return array|\Illuminate\Contracts\Support\Arrayable|\
    JsonSerializable
     */
    public function toArray($request)
    {
        return [
            'count_reports' => count($this->collection),
            'reports' => ShortReportHoursResource::collection($this-
>collection),
        ];
    }
}

```

```

    }
}

app/Http/Resources/ReportHours/ShortReportHoursResource.php
<?php

namespace App\Http\Resources\ReportHours;

use Illuminate\Http\Resources\Json\JsonResource;

class ShortReportHoursResource extends JsonResource
{
    /**
     * Transform the resource into an array.
     *
     * @param \Illuminate\Http\Request $request
     * @return array|\Illuminate\Contracts\Support\Arrayable|\
JsonSerializable
     */
    public function toArray($request)
    {
        return [
            'nick' => $this->resource->nick,
            'name' => $this->resource->name,
            'company' => $this->resource->company,
            'total_hours' => $this->resource->total_hours,
        ];
    }
}

```

app/Models/BlogRecord.php

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Kirschbaum\PowerJoins\PowerJoins;

class BlogRecord extends Model
{
    use HasFactory;
    use PowerJoins;

    protected $table = 'blog_record';
    public $timestamps = false;

    public function dailyBlog()
    {
        return $this->belongsTo(DailyBlog::class, 'blog_id');
    }

    public function project()
    {
        return $this->belongsTo(Project::class);
    }
}
```

app/Models/Customer.php

&lt;?php



```

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Kirschbaum\PowerJoins\PowerJoins;

class Customer extends Model
{
    use HasFactory;
    use PowerJoins;

    protected $table = 'customer';
    public $timestamps = false;

    public function projects()
    {
        return $this->hasMany(Project::class);
    }
}

```

app/Models/DailyBlog.php

```

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Kirschbaum\PowerJoins\PowerJoins;

class DailyBlog extends Model
{

```

```
use HasFactory;
```

```
use PowerJoins;
```

```
protected $table = 'daily_blog';
```

```
public $timestamps = false;
```

```
public function weeklyBlog()
```

```
{
```

```
    return $this->belongsTo(WeeklyBlog::class, 'week_blog_id');
```

```
}
```

```
public function blogRecords()
```

```
{
```

```
    return $this->hasMany(BlogRecord::class, 'blog_id');
```

```
}
```

```
}
```

```
app/Models/Department.php
```

```
<?php
```

```
namespace App\Models;
```

```
use Illuminate\Database\Eloquent\Factories\HasFactory;
```

```
use Illuminate\Database\Eloquent\Model;
```

```
use Kirschbaum\PowerJoins\PowerJoins;
```

```
class Department extends Model
```

```
{
```

```
    use HasFactory;
```

```
    use PowerJoins;
```

```

protected $table = 'department';
public $timestamps = false;

public function workers()
{
    return $this->hasMany(Worker::class);
}
}

app/Models/Project.php

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Kirschbaum\PowerJoins\PowerJoins;

class Project extends Model
{
    use HasFactory;
    use PowerJoins;

    protected $table = 'project';
    public $timestamps = false;

    public function customer()
    {
        return $this->belongsTo(Customer::class);
    }
}

```

```

public function blogRecords()
{
    return $this->hasMany(BlogRecord::class);
}
}

```

app/Models/ReportHours.php

```
<?php
```

```
namespace App\Models;
```

```
use Illuminate\Support\Facades\DB;
```

```
class ReportHours
```

```

{
    private $request;

    public function __construct(
        private $startDate = null,
        private $endDate = null
    ){
        $this->request = BlogRecord::joinRelationship('dailyBlog')
        ->select([
            DB::raw('SUM(`time`) as `total_hours`')
        ]);
        if ($this->startDate)
        {
            $this->request->where(
                'date',
                '>=',
                date_create($this->startDate)
            );
        }
    }
}

```

```

        );
    }
    if ($this->endDate)
    {
        $this->request->where(
            'date',
            '<=',
            date_create($this->endDate)
        );
    }
}

private function withWorker() {
    $this->request
        ->joinRelationship('dailyBlog.weeklyBlog.worker')
        ->addSelect([
            'worker.nick',
            'worker.name',
            'worker.company',
        ])
        ->groupBy('nick');
    return $this;
}

private function withProject() {
    $this->request
        ->joinRelationship('project.customer')
        ->addSelect([
            'project_id',
            'project.name as project_name',
            'customer_id',

```

```

        'customer.name as customer_name',
    ])
    ->groupBy('project.id');
    return $this;
}

public function reportShort()
{
    return $this->withWorker()->request->get();
}

public function reportByProject()
{
    return $this
        ->withWorker()
        ->withProject()
        ->request
        ->get();
}
}

```

app/Models/User.php

```
<?php
```

```
namespace App\Models;
```

```
use Illuminate\Contracts\Auth\MustVerifyEmail;
```

```
use Illuminate\Database\Eloquent\Factories\HasFactory;
```

```
use Illuminate\Foundation\Auth\User as Authenticatable;
```

```
use Illuminate\Notifications\Notifiable;
```

```
use Laravel\Sanctum\HasApiTokens;
```

```

use Kirschbaum\PowerJoins\PowerJoins;

class User extends Authenticatable
{
    use HasApiTokens, HasFactory, Notifiable;
    use PowerJoins;

    /**
     * The attributes that are mass assignable.
     *
     * @var array<int, string>
     */
    protected $fillable = [
        'name',
        'email',
        'password',
    ];

    /**
     * The attributes that should be hidden for serialization.
     *
     * @var array<int, string>
     */
    protected $hidden = [
        'password',
        'remember_token',
    ];

    /**
     * The attributes that should be cast.
     *

```

```

* @var array<string, string>
*/
protected $casts = [
    'email_verified_at' => 'datetime',
];

public function worker()
{
    return $this->hasOne(Worker::class, 'nick', 'nick');
}
}

```

app/Models/WeeklyBlog.php

```

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Kirschbaum\PowerJoins\PowerJoins;

class WeeklyBlog extends Model
{
    use HasFactory;
    use PowerJoins;

    protected $table = 'weekly_blog';
    public $timestamps = false;

    public function worker()
    {

```



```

        return $this->belongsTo(Worker::class);
    }

```

```

    public function dailyBlogs()
    {
        return $this->hasMany(DailyBlog::class, 'week_blog_id');
    }
}

```

app/Models/Worker.php

```
<?php
```

```

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Kirschbaum\PowerJoins\PowerJoins;

class Worker extends Model
{
    use HasFactory;
    use PowerJoins;

    protected $table = 'worker';
    protected $primaryKey = 'nick';
    protected $keyType = 'string';
    public $timestamps = false;

    public function user()
    {
        return $this->belongsTo(User::class, 'nick', 'nick');
    }
}

```

```
}
```

```
public function department()
{
    return $this->belongsTo(Department::class);
}
```

```
public function weeklyBlogs()
{
    return $this->hasMany(WeeklyBlog::class);
}
}
```

app/Operations/Schedule/Day.php

```
<?php
namespace App\Operations\Schedule;

use App\Operations\Schedule\Modifiers\Modifier;
use App\Operations\Schedule\Modifiers\ManagerModifier;

class Day
{
    private \DateTime $date;

    private array $modifiers = [];

    public function __construct(\DateTime $date)
    {
        $this->date = $date;
    }
}
```

```

public function addModifier(Modifier $modifier)
{
    $this->modifiers[] = $modifier;
}

public function getStatus()
{
    if(!$this->modifiers)
    {
        return 'work';
    }
    $manager = new ManagerModifier();
    return array_reduce(
        $this->modifiers,
        fn($cary, $item) => $manager->resolveTwoModifier($item, $cary),
        )->getStatus();
}

public function getInfo()
{
    return array_map(fn($mod)=>$mod->getInfo(), $this->modifiers);
}

public function getDate()
{
    return clone $this->date;
}
}

```

app/Operations/Schedule/GetterVacationOfficial.php

<?php

```

namespace App\Operations\Schedule;

use App\Models\VacationOfficial;

class GetterVacationOfficial
{
    public Schedule $schedule;

    public function __construct(RangeDate $range)
    {
        $this->schedule = new Schedule($range);
        foreach (VacationOfficial::getAllRangeModifiersInRange($range) as
$rangeModifier)
        {
            $this->schedule->applyRangeModifier($rangeModifier);
        }
    }
}

```

app/Operations/Schedule/GetterWorkerVacation.php

```

<?php
namespace App\Operations\Schedule;

use App\Models\Vacation;
use App\Models\Worker;
use App\Models\ReportHours;

class GetterWorkerVacation
{
    private RangeDate $range;

```

```

public Schedule $officialSchedule;

public array $workersSchedule = [];

public function __construct(RangeDate $range)
{
    $this->range = $range;
    $this->officialSchedule = (new GetterVacationOfficial($range))
        ->schedule;
    $workersChanges =
Vacation::getAllWorkersRangeModifiersInRange($range);
    foreach ($workersChanges as $nick => $rangeModifiers)
    {
        $this->workersSchedule[$nick] = $this->officialSchedule->clone();
        foreach ($rangeModifiers as $rangeModifier)
        {
            $this->workersSchedule[$nick]-
>applyRangeModifier($rangeModifier);
        }
    }
}

public function getWorkersShedule(array $nicks)
{
    $result = [];
    foreach ($nicks as $nick)
    {
        $result[$nick] = $this->getSheduleForWorker($nick);
    }
    return $result;
}

```

```

public function getSheduleForWorker(string $nick)
{
    return $this->workersSchedule[$nick] ?? $this->officialSchedule-
>clone();
}

public function getScheduleWithWorkersHours()
{
    $schedules = $this-
>getWorkersShedule(Worker::getNicksEnabledWorkers());
    $schedules = array_map(function($schedule){
        return array_map(function($day){
            return [
                'status' => $day->getStatus(),
                'info' => $day->getInfo(),
            ];
        },$schedule->getDays());
    },$schedules);
    $workersHours = new ReportHours(
        $this->range->getStart()->format('Y-m-d'),
        $this->range->getEnd()->format('Y-m-d'),
    );
    foreach ($workersHours->reportByDate() as $record)
    {
        $schedules[$record->nick][$record->date]['total_hours'] = $record-
>total_hours;
    }
    return $schedules;
}
}

```

## app/Operations/Schedule/Modifiers/Conflict.php

```
<?php
namespace App\Operations\Schedule\Modifiers;

use App\Operations\Schedule\Modifiers\Modifier;

class Conflict implements Modifier
{
    private string $info;

    public function __construct(string $info)
    {
        $this->info = $info;
    }

    public function getInfo(): string
    {
        return $this->info;
    }

    public function getStatus(): ?string
    {
        return 'conflict';
    }

    public function getType(): string
    {
        return 'conflict';
    }
}
```

app/Operations/Schedule/Modifiers/Holiday.php

```
<?php
namespace App\Operations\Schedule\Modifiers;

use App\Operations\Schedule\Modifiers\Modifier;

class Holiday implements Modifier
{
    private string $info;

    public function __construct(string $info)
    {
        $this->info = $info;
    }

    public function getInfo(): string
    {
        return $this->info;
    }

    public function getStatus(): ?string
    {
        return 'holiday';
    }

    public function getType(): string
    {
        return 'holiday';
    }
}
```



app/Operations/Schedule/Modifiers/ManagerModifier.php

```
<?php
```

```
namespace App\Operations\Schedule\Modifiers;
```

```
class ManagerModifier
```

```
{
```

```
    private array $dictionaryType = [];
```

```
    private function getAllModifierWithPriority(): array
```

```
    {
```

```
        return [
```

```
            Holiday::class => 10,
```

```
            Official::class => 20,
```

```
            Real::class => 40,
```

```
            Work::class => 40,
```

```
            Conflict::class => 10000,
```

```
        ];
```

```
    }
```

```
    public function __construct()
```

```
    {
```

```
        $this->dictionaryType = [];
```

```
        foreach ($this->getAllModifierWithPriority() as $class => $priority)
```

```
        {
```

```
            $modifier = new $class("");
```

```
            $this->dictionaryType[$modifier->getType()] = [
```

```
                'class' => $class,
```

```
                'priority' => $priority
```

```
            ];
```

```
            unset($modifier);
```

```

    }
}

public function createModifierByType(string $type, string $info)
{
    if(!isset($this->dictionaryType[$type]))
    {
        throw new \Exception("Type day $type not found");
    }
    return new $this->dictionaryType[$type]['class']($info);
}

public function resolveTwoModifier(Modifier $m1, ?Modifier $m2)
{
    if(!$m2)
    {
        return $m1;
    }
    $priority1 = $this->dictionaryType[$m1->getType()]['priority'];
    $priority2 = $this->dictionaryType[$m2->getType()]['priority'];
    if($priority1 < $priority2)
    {
        return $m2;
    }
    if($priority1 > $priority2)
    {
        return $m1;
    }
    if($m1->getStatus() !== $m2->getStatus())
    {
        return new Conflict(

```

```
"Conflict {$m1->getType()} and {$m2->getType()}."
```

```
);
```

```
}
```

```
}
```

```
}
```

```
app/Operations/Schedule/Modifiers/Modifier.php
```

```
<?php
```

```
namespace App\Operations\Schedule\Modifiers;
```

```
interface Modifier
```

```
{
```

```
    public function getStatus(): ?string;
```

```
    public function getInfo(): string;
```

```
    public function getType(): string;
```

```
}
```

```
app/Operations/Schedule/Modifiers/Official.php
```

```
<?php
```

```
namespace App\Operations\Schedule\Modifiers;
```

```
use App\Operations\Schedule\Modifiers\Modifier;
```

```
class Official implements Modifier
```

```
{
```

```
    private string $info;
```

```
    public function __construct(string $info)
```

```
    {
```

```
        $this->info = $info;
```

```
    }
```

```

public function getInfo(): string
{
    return $this->info;
}

```

```

public function getStatus(): ?string
{
    return null;
}

```

```

public function getType(): string
{
    return 'Official';
}
}

```

app/Operations/Schedule/Modifiers/Real.php

```
<?php
```

```
namespace App\Operations\Schedule\Modifiers;
```

```
use App\Operations\Schedule\Modifiers\Modifier;
```

```
class Real implements Modifier
```

```

{
    private string $info;

```

```

    public function __construct(string $info)
    {
        $this->info = $info;
    }

```

```

public function getInfo(): string
{
    return $this->info;
}

```

```

public function getStatus(): ?string
{
    return 'holiday';
}

```

```

public function getType(): string
{
    return 'Real';
}
}

```

app/Operations/Schedule/Modifiers/Work.php

```

<?php
namespace App\Operations\Schedule\Modifiers;

use App\Operations\Schedule\Modifiers\Modifier;

class Work implements Modifier
{
    private string $info;

    public function __construct(string $info)
    {
        $this->info = $info;
    }
}

```

```

public function getInfo(): string
{
    return $this->info;
}

```

```

public function getStatus(): ?string
{
    return 'work';
}

```

```

public function getType(): string
{
    return 'Work';
}
}

```

app/Operations/Schedule/RangeDate.php

```

<?php
namespace App\Operations\Schedule;

use DateTime;

class RangeDate
{
    private DateTime $start;
    private DateTime $end;

    public function __construct(
        DateTime $start,
        DateTime $end
    )
    {
    }
}

```

```

) {
    $this->start = $start;
    $this->end = $end;
    $this->checkStartDate();
}

public static function createFromStrings(
    string $start_date,
    string $end_date
): self {
    return new RangeDate(
        new \DateTime($start_date),
        new \DateTime($end_date)
    );
}

public function getStart(): DateTime
{
    return clone $this->start;
}

public function getEnd(): DateTime
{
    return clone $this->end;
}

public function setStart(DateTime $start): void
{
    $this->start = $start;
    $this->checkStartDate();
}

```

```

public function setEnd(DateTime $end): void
{
    $this->end = $end;
    $this->checkStartDate();
}

```

```

private function checkStartDate(): void
{
    if ($this->start > $this->end)
    {
        $this->exceptionStartAfterEnd();
    }
}

```

```

private function exceptionStartAfterEnd(): void
{
    $startStr = $this->start->format('Y-n-j G:i:s');
    $endStr = $this->end->format('Y-n-j G:i:s');
    throw new \Exception(
        "Start ($startStr) must be before end ($endStr)."
    );
}

```

```

public function intersection(self $range): self
{
    $start = max($this->start, $range->start);
    $end = min($this->end, $range->end);
    return new self($start, $end);
}

```



```

public function iterator(): \Generator
{
    for (
        $date = clone $this->start;
        $date <= $this->end;
        $date->add(new \DateInterval('P1D'))
    ) {
        yield $date;
    }
}
}

app/Operations/Schedule/RangeDaysModifier.php

<?php
namespace App\Operations\Schedule;

use App\Operations\Schedule\Modifiers\Modifier;
use App\Operations\Schedule\Modifiers\ManagerModifier;

class RangeDaysModifier
{
    public RangeDate $range;

    public Modifier $modifier;

    static function create(
        string $start_date,
        string $end_date,
        string $type,
        string $info
    ): self

```

```

{
    $rdm = new self();
    $rdm->range = RangeDate::createFromStrings($start_date, $end_date);
    $manager = new ManagerModifier();
    $rdm->modifier = $manager->createModifierByType($type, $info);
    return $rdm;
}
}

```

app/Operations/Schedule/Schedule.php

```

<?php
namespace App\Operations\Schedule;

use App\Operations\Schedule\RangeDaysModifier;

class Schedule
{
    private $range;
    private $days = [];

    public function __construct(RangeDate $range)
    {
        $this->range = $range;
        $this->initDays();
    }

    private function initDays()
    {
        foreach ($this->range->iterator() as $day)
        {
            $this->days[$day->format('Y-m-d')] = new Day($day);
        }
    }
}

```

```

    }
}

    public function applyRangeModifier(RangeDaysModifier
$rangeModifier)
    {
        $range = $rangeModifier->range->intersection($this->range);
        foreach ($range->iterator() as $day)
        {
            $this->days[$day->format('Y-m-d')]
                ->addModifier($rangeModifier->modifier);
        }
    }

    public function getDays()
    {
        return $this->days;
    }

    public function clone(): self
    {
        $newSchedule = new self($this->range);
        $newSchedule->days = array_map(fn($day)=>clone $day, $this-
>days);
        return $newSchedule;
    }
}

composer.json

{
    "name": "laravel/laravel",

```

```

"type": "project",
"description": "The Laravel Framework.",
"keywords": ["framework", "laravel"],
"license": "MIT",
"require": {
    "php": "^8.0.2",
    "adldap2/adldap2-laravel": "^6.1",
    "fruitcake/laravel-cors": "^2.0.5",
    "guzzlehttp/guzzle": "^7.2",
    "kirschbaum-development/eloquent-power-joins": "^2.6.2",
    "laravel/framework": "^9.0",
    "laravel/sanctum": "^2.14",
    "laravel/tinker": "^2.7"
},
"require-dev": {
    "fakerphp/faker": "^1.9.1",
    "knuckleswtf/scribe": "^3.24",
    "laravel/sail": "^1.0.1",
    "mockery/mockery": "^1.4.4",
    "nunomaduro/collision": "^6.1",
    "phpunit/phpunit": "^9.5.10",
    "spatie/laravel-ignition": "^1.0"
},
"autoload": {
    "psr-4": {
        "App\\": "app/",
        "Database\\Factories\\": "database/factories/",
        "Database\\Seeders\\": "database/seeders/"
    }
},
"autoload-dev": {

```

```

    "psr-4": {
        "Tests\\": "tests/"
    }
},
"scripts": {
    "post-autoload-dump": [
        "Illuminate\\Foundation\\ComposerScripts::postAutoloadDump",
        "@php artisan package:discover --ansi"
    ],
    "post-update-cmd": [
        "@php artisan vendor:publish --tag=laravel-assets --ansi --force"
    ],
    "post-root-package-install": [
        "@php -r \"file_exists('.env') || copy('.env.example', '.env');\""
    ],
    "post-create-project-cmd": [
        "@php artisan key:generate --ansi"
    ]
},
"extra": {
    "laravel": {
        "dont-discover": []
    }
},
"config": {
    "optimize-autoloader": true,
    "preferred-install": "dist",
    "sort-packages": true
},
"minimum-stability": "dev",
"prefer-stable": true

```

```
}
```

```
database/factories/BlogRecordFactory.php
```

```
<?php
```

```
namespace Database\Factories;
```

```
use Illuminate\Database\Eloquent\Factories\Factory;
```

```
use App\Models\DailyBlog;
```

```
use App\Models\Project;
```

```
/**
```

```
 * @extends \Illuminate\Database\Eloquent\Factories\Factory<\App\Models\
BlogRecord>
```

```
 */
```

```
class BlogRecordFactory extends Factory
```

```
{
```

```
 /**
```

```
 * Define the model's default state.
```

```
 *
```

```
 * @return array
```

```
 */
```

```
public function definition()
```

```
{
```

```
    return [
```

```
        'blog_id' => DailyBlog::factory(),
```

```
        'project_id' => Project::factory(),
```

```
        'time' => rand(0,100)/10,
```

```
        'type' => $this->faker->randomElement([
```

```
            'Bug Fixing',
```

```
            'Development',
```

```

        'Support',
        'Consulting',
        'Testing',
        'Design',
        'Plan',
        'Documentation',
        'Localization'
    ),
    'description' => $this->faker->text(),
    'plan_notify' => $this->faker->randomElement(['0', '1']),
    'plan_notify_date' => $this->faker->date(),
    'dt' => $this->faker->dateTime(),
];
}
}

```

database/factories/CustomerFactory.php

```
<?php
```

```
namespace Database\Factories;
```

```
use Illuminate\Database\Eloquent\Factories\Factory;
```

```
/**
```

```
 * @extends \Illuminate\Database\Eloquent\Factories\Factory<\App\Models\
Customer>
```

```
 */
```

```
class CustomerFactory extends Factory
```

```
{
```

```
/**
```

```
 * Define the model's default state.
```

```

*
* @return array
*/
public function definition()
{
    return [
        'name' => $this->faker->words(asText: true),
        'contact' => 'email: ' . $this->faker->email()
    ];
}
}

database/factories/DailyBlogFactory.php

<?php

namespace Database\Factories;

use Illuminate\Database\Eloquent\Factories\Factory;
use App\Models\WeeklyBlog;

/**
 * @extends \Illuminate\Database\Eloquent\Factories\Factory<\App\Models\
DailyBlog>
 */
class DailyBlogFactory extends Factory
{
    /**
     * Define the model's default state.
     *
     * @return array
     */

```



```

public function definition()
{
    return [
        'week_blog_id' => WeeklyBlog::factory(),
        'date' => $this->faker->date(),
        'filled' => rand(0, 127),
    ];
}
}

```

database/factories/DepartmentFactory.php

```
<?php
```

```
namespace Database\Factories;
```

```
use Illuminate\Database\Eloquent\Factories\Factory;
```

```
/**
```

```
 * @extends \Illuminate\Database\Eloquent\Factories\Factory<\App\Models\
Department>
```

```
 */
```

```
class DepartmentFactory extends Factory
```

```
{
```

```
 /**
```

```
 * Define the model's default state.
```

```
 *
```

```
 * @return array
```

```
 */
```

```
public function definition()
```

```
{
```

```
    return [
```

```

        'name' => $this->faker->words(asText: true),
        'description' => $this->faker->text(),
        'ordinal' => rand(0, 65535),
    ];
}
}

```

database/factories/ProjectFactory.php

```
<?php
```

```
namespace Database\Factories;
```

```
use Illuminate\Database\Eloquent\Factories\Factory;
```

```
use App\Models\Customer;
```

```
use App\Models\Project;
```

```
/**
```

```

 * @extends \Illuminate\Database\Eloquent\Factories\Factory<\App\Models\
Project>

```

```
 */
```

```
class ProjectFactory extends Factory
```

```
{
```

```
    /**
```

```
     * Define the model's default state.
```

```
     *
```

```
     * @return array
```

```
     */
```

```
public function definition()
```

```
{
```

```
    return [
```

```
        'name' => $this->faker->words(asText: true),
```

```

        'description' => $this->faker->text(),
        'status' => $this->faker->randomElement([
            'investigation',
            'development',
            'support',
            'frozen',
            'closed'
        ]),
        'customer_id' => Customer::factory(),
        'parent_id' => null,
        'email' => $this->faker->email(),
        'reportable' => rand(0, 127),
        'report_day' => $this->faker->dayOfWeek(),
        'report_time' => $this->faker->time(),
        'note' => $this->faker->text(),
    ];
}
}

```

database/factories/UserFactory.php

```
<?php
```

```
namespace Database\Factories;
```

```
use Illuminate\Database\Eloquent\Factories\Factory;
```

```
use App\Models\User;
```

```
/**
```

```

 * @extends \Illuminate\Database\Eloquent\Factories\Factory<\App\Models\
User>
 */

```

```

class UserFactory extends Factory
{
    protected $model = User::class;

    /**
     * Define the model's default state.
     *
     * @return array<string, mixed>
     */
    public function definition()
    {
        return [
            'nick' => substr($this->faker->userName(),0,15),
            'password' => $this->faker->sha256(),
        ];
    }
}

```

database/factories/VacationFactory.php

```
<?php
```

```
namespace Database\Factories;
```

```
use Carbon\Carbon;
```

```
use App\Models\Worker;
```

```
use Illuminate\Database\Eloquent\Factories\Factory;
```

```
/**
```

```

 * @extends \Illuminate\Database\Eloquent\Factories\Factory<\App\Models\
Vacation>
 */

```

```

class VacationFactory extends Factory
{
    /**
     * Define the model's default state.
     *
     * @return array
     */
    public function definition()
    {
        return [
            'worker_nick' => fn()=>Worker::factory()->create()->nick,
            'start_date' => $start_date = $this->faker->date(),
            'end_date' => (new Carbon($start_date))->addDay(rand(0,10)),
            'type' => $this->faker->randomElement(['Official','Real','Work']),
            'comment' => $this->faker->sentence(),
        ];
    }
}

```

database/factories/VacationOfficialFactory.php

```
<?php
```

```
namespace Database\Factories;
```

```
use Carbon\Carbon;
```

```
use App\Models\Worker;
```

```
use Illuminate\Database\Eloquent\Factories\Factory;
```

```
/**
```

```

 * @extends \Illuminate\Database\Eloquent\Factories\Factory<\App\Models\
Vacation>

```

```

*/
class VacationOfficialFactory extends Factory
{
    /**
     * Define the model's default state.
     *
     * @return array
     */
    public function definition()
    {
        return [
            'start_date' => $start_date = $this->faker->date(),
            'end_date' => (new Carbon($start_date))->addDay(rand(0,10)),
            'comment' => $this->faker->sentence(),
        ];
    }
}

```

database/factories/WeeklyBlogFactory.php

```
<?php
```

```
namespace Database\Factories;
```

```
use Illuminate\Database\Eloquent\Factories\Factory;
```

```
use App\Models\Worker;
```

```
/**
```

```
 * @extends \Illuminate\Database\Eloquent\Factories\Factory<\App\Models\
WeeklyBlog>

```

```
*/
```

```
class WeeklyBlogFactory extends Factory
```

```

{
    /**
     * Define the model's default state.
     *
     * @return array
     */
    public function definition()
    {
        return [
            'worker_nick' => fn()=>Worker::factory()->create()->nick,
            'start_date' => $this->faker->date(),
            'end_date' => $this->faker->date(),
        ];
    }
}

```

database/factories/WorkerFactory.php

```
<?php
```

```
namespace Database\Factories;
```

```
use Illuminate\Database\Eloquent\Factories\Factory;
```

```
use App\Models\Department;
```

```
/**
```

```

 * @extends \Illuminate\Database\Eloquent\Factories\Factory<\App\Models\
Worker>

```

```
*/
```

```
class WorkerFactory extends Factory
```

```
{
```

```
/**
```

```
* Define the model's default state.
```

```
*
```

```
* @return array
```

```
*/
```

```
public function definition()
```

```
{
```

```
    return [
```

```
        'nick' => substr($this->faker->userName(),0,15),
```

```
        'name' => $this->faker->name(),
```

```
        'birthday' => $this->faker->date(max: '2006-01-01'),
```

```
        'role' => $this->faker->words(asText: true),
```

```
        'email' => $this->faker->email(),
```

```
        'department_id' => Department::factory(),
```

```
        'department_lead' => rand(0, 255),
```

```
        'image_small' => "",
```

```
        'image_big' => "",
```

```
        'old_domain_pass' => $this->faker->password(),
```

```
        'enabled' => rand(0, 255),
```

```
        'company' => $this->faker->randomElement(['TC', 'P'])
```

```
    ];
```

```
}
```

```
}
```

```
database/migrations/2014_10_12_000000_create_users_table.php
```

```
<?php
```

```
use Illuminate\Database\Migrations\Migration;
```

```
use Illuminate\Database\Schema\Blueprint;
```

```
use Illuminate\Support\Facades\Schema;
```

```
return new class extends Migration
```



```

{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('users', function (Blueprint $table) {
            $table->id();

            $table->string('objectguid')->nullable(); // For Adldap2.
            $table->string('nick', 15)->unique();
            $table->string('password');

            $table->rememberToken();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('users');
    }
};

```

database/migrations/2022\_02\_20\_082301\_create\_departments\_table.php

```
<?php
```

```
use Illuminate\Database\Migrations\Migration;
```

```
use Illuminate\Database\Schema\Blueprint;
```

```
use Illuminate\Support\Facades\Schema;
```

```
return new class extends Migration
```

```
{
```

```
    /**
```

```
     * Run the migrations.
```

```
     *
```

```
     * @return void
```

```
    */
```

```
    public function up()
```

```
    {
```

```
        Schema::create('department', function (Blueprint $table) {
```

```
            $table->id();
```

```
            $table->string('name', 255)->default("");
```

```
            $table->string('description', 255)->nullable();
```

```
                $table->smallInteger('ordinal', unsigned: true)->default(0)-
```

```
>nullable();
```

```
        });
```

```
    }
```

```
    /**
```

```
     * Reverse the migrations.
```

```
     *
```

```
     * @return void
```

```
    */
```

```

public function down()
{
    Schema::dropIfExists('department');
}
};

```

database/migrations/2022\_02\_20\_101240\_create\_worker\_table.php

```
<?php
```

```

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

```

```
return new class extends Migration
```

```

{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('worker', function (Blueprint $table) {
            $table->string('nick', 15)->primary();
            $table->string('name', 255)->default("");
            $table->date('birthday')->nullable();
            $table->string('role', 255)->nullable();
            $table->string('email', 50)->nullable();
            $table->smallInteger('department_id', unsigned: true)->nullable();
            $table->tinyInteger('department_lead', unsigned: true)
                ->default('0')

```

```

        ->nullable();
        $table->string('image_small', 255)->nullable();
        $table->string('image_big', 255)->nullable();
        $table->string('old_domain_pass', 40)->nullable();
        $table->tinyInteger('enabled', unsigned: true)->default(1);
        $table->string('company', 10)->default("");
    });
}

/**
 * Reverse the migrations.
 *
 * @return void
 */
public function down()
{
    Schema::dropIfExists('worker');
}
};

database/migrations/2022_02_20_101254_create_project_table.php

<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.

```

```

*
* @return void
*/
public function up()
{
    Schema::create('project', function (Blueprint $table) {
        $table->id();
        $table->string('name', 255)->default("");
        $table->text('description')->nullable();
        $table->enum('status', [
            'investigation',
            'development',
            'support',
            'frozen',
            'closed'
        ]->default('investigation'));
        $table->integer('customer_id', unsigned: true)->default(0);
        $table->integer('parent_id', unsigned: true)->nullable();
        $table->string('email', 100)->nullable();
        $table->tinyInteger('reportable')->default(0)->nullable();
        $table->string('report_day', 10)->nullable();
        $table->time('report_time')->nullable();
        $table->text('note')->nullable();
    });
}

/**
 * Reverse the migrations.
 *
 * @return void
 */

```

```

public function down()
{
    Schema::dropIfExists('project');
}
};

```

database/migrations/2022\_02\_20\_101257\_create\_weekly\_blog\_table.php

```
<?php
```

```

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;
use Carbon\Carbon;

```

```
return new class extends Migration
```

```

{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('weekly_blog', function (Blueprint $table) {
            $table->id();
            $table->string('worker_nick', 15);
            $table->date('start_date')->default(Carbon::create());
            $table->date('end_date')->default(Carbon::create());
            $table->unique(['start_date', 'end_date', 'worker_nick'], 'uniq');
        });
    }
}

```

```

/**
 * Reverse the migrations.
 *
 * @return void
 */
public function down()
{
    Schema::dropIfExists('weekly_blog');
}
};

database/migrations/2022_02_20_101300_create_daily_blog_table.php

<?php

use Carbon\Carbon;
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('daily_blog', function (Blueprint $table) {
            $table->id();

```

```

        $table->integer('week_blog_id', unsigned: true)->default(0);
        $table->date('date')->default(Carbon::create());
        $table->tinyInteger('filled')->default(0)->nullable();
        $table->unique(['date', 'week_blog_id'], 'date');
    });
}

/**
 * Reverse the migrations.
 *
 * @return void
 */
public function down()
{
    Schema::dropIfExists('daily_blog');
}
};

```

database/migrations/2022\_02\_20\_101303\_create\_blog\_record\_table.php

```
<?php
```

```

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

```

```
return new class extends Migration
```

```

{
    /**
     * Run the migrations.
     *
     * @return void
     */
}

```



```

*/
public function up()
{
    Schema::create('blog_record', function (Blueprint $table) {
        $table->id();
        $table->integer('blog_id', unsigned: true)->default(0);
        $table->integer('project_id', unsigned: true)->default(0);
        $table->float('time', 3, 1)->default(0);
        $table->enum('type', [
            'Bug Fixing',
            'Development',
            'Support',
            'Consulting',
            'Testing',
            'Design',
            'Plan',
            'Documentation',
            'Localization'
        ]->nullable());
        $table->text('description') ;
        $table->enum('plan_notify', ['0','1'])->default('0')->nullable();
        $table->date('plan_notify_date')->nullable();
        $table->dateTime('dt')->nullable();
    });
}

/**
 * Reverse the migrations.
 *
 * @return void
 */

```

```

public function down()
{
    Schema::dropIfExists('blog_record');
}
};

database/migrations/2022_02_20_104038_create_customer_table.php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('customer', function (Blueprint $table) {
            $table->id();
            $table->string('name', 255)->default("");
            $table->string('contact', 255)->nullable();
        });
    }

    /**
     * Reverse the migrations.

```

```

*
* @return void
*/
public function down()
{
    Schema::dropIfExists('customer');
}
};

```

database/migrations/2022\_04\_21\_021010\_create\_vacation\_official\_table.php

```

<?php

use Carbon\Carbon;
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('vacation_official', function (Blueprint $table) {
            $table->id();
            $table->date('start_date')->default(Carbon::create());
            $table->date('end_date')->default(Carbon::create());
            $table->string('comment', 255)->nullable()->default("");
        });
    }
};

```

```

    });
}

/**
 * Reverse the migrations.
 *
 * @return void
 */
public function down()
{
    Schema::dropIfExists('vacation_official');
}
};

database/migrations/2022_04_21_021015_create_vacation_table.php

<?php

use Carbon\Carbon;
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {

```

```

Schema::create('vacation', function (Blueprint $table) {
    $table->id();
    $table->string('worker_nick', 15);
    $table->date('start_date')->default(Carbon::create());
    $table->date('end_date')->default(Carbon::create());
    $table->string('type', 50)->default("");
    $table->string('comment', 255)->nullable()->default(null);
});
}

/**
 * Reverse the migrations.
 *
 * @return void
 */
public function down()
{
    Schema::dropIfExists('vacation');
}
};

```

database/seeder/DatabaseSeeder.php

```
<?php
```

```
namespace Database\Seeders;
```

```
use Illuminate\Database\Seeder;
```

```
use Database\Seeders\usual\UsualDatabaseSeeder;
```

```
class DatabaseSeeder extends Seeder
```

```
{
```

```

/**
 * Seed the application's database.
 *
 * @return void
 */
public function run()
{
    $this->call([
        UsualDatabaseSeeder::class
    ]);
}
}

database/seeder/ForDocument/ForDocumentSeeder.php

<?php

namespace Database\Seeders\ForDocument;

use App\Models\BlogRecord;
use App\Models\DailyBlog;
use App\Models\User;
use Illuminate\Database\Seeder;
use Illuminate\Support\Facades\Hash;

class ForDocumentSeeder extends Seeder
{
    public function run()
    {
        $this->createUser();
        $this->createBlogs();
    }
}

```

```

private function createUser()
{
    $nick = env('SCRIBE_USER_NICK', 'scribe');
    User::where('nick', $nick)->delete();
    $user = User::factory()->create([
        'nick' => $nick,
        'password' => Hash::make(env('SCRIBE_USER_PASSWORD',
'password'))),
    ]);
    $token = $user->createToken('report-service')->plainTextToken;
    $this->setenv('SCRIBE_AUTH_KEY', $token);
}

```

```

private function setenv($key, $value)
{
    $path = app()->environmentFilePath();
    file_put_contents(
        $path,
        preg_replace(
            '/' . $key . '=.*/m',
            $key . '=' . $value,
            file_get_contents($path)
        )
    );
}

```

```

private function createBlogs()
{
    $dailyBlog = DailyBlog::factory()->create(['date' => '2010-10-08']);
    BlogRecord::factory(2)->create(['blog_id' => $dailyBlog]);
}

```

```

    }
}

```

database/seeder/usual/BlogRecordSeeder.php

```
<?php
```

```
namespace Database\Seeders\usual;
```

```
use App\Models\BlogRecord;
```

```
use Illuminate\Database\Seeder;
```

```
class BlogRecordSeeder extends Seeder
```

```

{
    /**
     * Run the database seeds.
     *
     * @return void
     */
    public function run()
    {
        BlogRecord::factory(30)->create();
    }
}

```

database/seeder/usual/CustomerSeeder.php

```
<?php
```

```
namespace Database\Seeders\usual;
```

```
use App\Models\Customer;
```

```
use Illuminate\Database\Seeder;
```



```

class CustomerSeeder extends Seeder
{
    /**
     * Run the database seeds.
     *
     * @return void
     */
    public function run()
    {
        Customer::factory(30)->create();
    }
}

```

database/seeder/usual/DailyBlogSeeder.php

```
<?php
```

```
namespace Database\Seeders\usual;
```

```
use App\Models\DailyBlog;
```

```
use Illuminate\Database\Seeder;
```

```

class DailyBlogSeeder extends Seeder
{
    /**
     * Run the database seeds.
     *
     * @return void
     */
    public function run()
    {

```

```

        DailyBlog::factory(30)->create();
    }
}

```

database/seeder/usual/DepartmentSeeder.php

```
<?php
```

```
namespace Database\Seeders\usual;
```

```
use App\Models\Department;
```

```
use Illuminate\Database\Seeder;
```

```
class DepartmentSeeder extends Seeder
```

```

{
    /**
     * Run the database seeds.
     *
     * @return void
     */
    public function run()
    {
        Department::factory(10)->create();
    }
}

```

database/seeder/usual/ProjectSeeder.php

```
<?php
```

```
namespace Database\Seeders\usual;
```

```
use App\Models\Project;
```

```
use Illuminate\Database\Seeder;
```

```
class ProjectSeeder extends Seeder
```

```
{
    public function run()
    {
        /*
         * To create a hierarchical structure of projects, you need to have
         * something to tie projects to. Therefore, they need to be created
         * not en masse, but one at a time.
         */
        for($i = 0; $i < 30; $i++)
        {
            Project::factory()->create();
        }
    }
}
```

database/seeder/usual/UserSeeder.php

```
<?php
```

```
namespace Database\Seeders\usual;
```

```
use App\Models\User;
```

```
use Illuminate\Database\Seeder;
```

```
class UserSeeder extends Seeder
```

```
{
    /**
     * Run the database seeds.
     */
}
```

```

    * @return void
    */
    public function run()
    {
        User::factory(30)->create();
    }
}

```

database/seeder/usual/UsualDatabaseSeeder.php

```
<?php
```

```
namespace Database\Seeders\usual;
```

```
use Illuminate\Database\Seeder;
```

```
class UsualDatabaseSeeder extends Seeder
```

```

{
    /**
     * Seed the application's database.
     *
     * @return void
     */
    public function run()
    {
        $this->call([
            UserSeeder::class,
            DepartmentSeeder::class,
            CustomerSeeder::class,
            WorkerSeeder::class,
            ProjectSeeder::class,
            WeeklyBlogSeeder::class,

```

```

        DailyBlogSeeder::class,
        BlogRecordSeeder::class,
        VacationOfficialSeeder::class,
        VacationSeeder::class,
    ];
}
}

database/seeder/usual/VacationOfficialSeeder.php

<?php

namespace Database\Seeders\usual;

use Illuminate\Database\Seeder;
use App\Models\VacationOfficial;

class VacationOfficialSeeder extends Seeder
{
    /**
     * Run the database seeds.
     *
     * @return void
     */
    public function run()
    {
        VacationOfficial::factory(30)->create();
    }
}

database/seeder/usual/VacationSeeder.php

<?php

```

```
namespace Database\Seeders\usual;
```

```
use Illuminate\Database\Seeder;
```

```
use App\Models\Vacation;
```

```
class VacationSeeder extends Seeder
```

```
{
```

```
    /**
```

```
     * Run the database seeds.
```

```
     *
```

```
     * @return void
```

```
    */
```

```
    public function run()
```

```
    {
```

```
        Vacation::factory(30)->create();
```

```
    }
```

```
}
```

```
database/seeders/usual/WeeklyBlogSeeder.php
```

```
<?php
```

```
namespace Database\Seeders\usual;
```

```
use App\Models\WeeklyBlog;
```

```
use Illuminate\Database\Seeder;
```

```
class WeeklyBlogSeeder extends Seeder
```

```
{
```

```
    /**
```

```
     * Run the database seeds.
```

```

*
* @return void
*/
public function run()
{
    WeeklyBlog::factory(30)->create();
}
}

```

database/seeder/usual/WorkerSeeder.php

```
<?php
```

```
namespace Database\Seeders\usual;
```

```
use Illuminate\Database\Console\Seeds\WithoutModelEvents;
```

```
use Illuminate\Database\Seeder;
```

```
use App\Models\Worker;
```

```
use App\Models\User;
```

```
class WorkerSeeder extends Seeder
```

```

{
    /**
     * Run the database seeds.
     *
     * @return void
     */
    public function run()
    {
        foreach (User::all() as $user)
        {
            Worker::factory()->state(['nick' => $user->nick])->create();

```

```

    }
}
}

```

routes/api.php

```
<?php
```

```

use App\Http\Actions\ReportHours\ActionShowReportHours;
use App\Http\Actions\ReportHours\ActionShowReportHoursByProject;
use App\Http\Actions\Schedule\ActionScheduleOfficial;
use App\Http\Actions\Schedule\ActionWorkersSchedule;
use App\Http\Actions\Token\Create\ActionCreateToken;
use App\Http\Actions\Token\Delete\ActionDeleteOneToken;
use App\Http\Actions\Token\Read\ActionReadAllTokens;
use App\Http\Actions\Token\Read\ActionReadOneToken;
use App\Http\Actions\Token\Update\ActionUpdateOneToken;
use App\Http\Actions\User\ActionLogin;
use Illuminate\Support\Facades\Route;

```

```
/*
```

```
|-----
```

```
| API Routes
```

```
|-----
```

```
|
```

```

| Here is where you can register API routes for your application. These
| routes are loaded by the RouteServiceProvider within a group which
| is assigned the "api" middleware group. Enjoy building your API!

```

```
|
```

```
*/
```

```
Route::post('login', ActionLogin::class);
```



```

Route::middleware('auth:sanctum')->group(function () {
    Route::post('token', ActionCreateToken::class);
    Route::get('token', ActionReadAllTokens::class);
    Route::get('token/{token_name}', ActionReadOneToken::class);
    Route::put('token/{token_name}', ActionUpdateOneToken::class);
    Route::delete('token/{token_name}', ActionDeleteOneToken::class);

    Route::get('report/hours/short', ActionShowReportHours::class);
    Route::get('report/hours/project',
ActionShowReportHoursByProject::class);

    Route::get('shedule/official', ActionScheduleOfficial::class);
    Route::get('shedule/worker', ActionWorkersSchedule::class);
});

```

tests/CreatesApplication.php

```
<?php
```

```
namespace Tests;
```

```
use Illuminate\Contracts\Console\Kernel;
```

```
trait CreatesApplication
```

```

{
    /**
     * Creates the application.
     *
     * @return \Illuminate\Foundation\Application
     */
    public function createApplication()

```

```

{
    $app = require __DIR__.'../bootstrap/app.php';

    $app->make(Kernel::class)->bootstrap();

    return $app;
}

```

tests/Feature/AuthTest.php

```
<?php
```

```
namespace Tests\Feature;
```

```
use Tests\TestCase;
```

```
/**
```

```
 * Tests for checking the availability of resources depending
```

```
 * on the authentication in the system.
```

```
 */
```

```
class AuthTest extends TestCase
```

```
{
```

```
    /**
```

```
     * Checking response codes in the absence of authentication.
```

```
     *
```

```
     * @return void
```

```
     */
```

```
    public function test_unauthorized()
```

```
    {
```

```
        $this->post('/api/login', [], [
```

```
            'Accept' => 'application/json'
```

```

    ]->assertStatus(422);

    $this->get('/api/report/hours/short')->assertStatus(401);
    $this->get('/api/report/hours/project')->assertStatus(401);

    $this->post('/api/token')->assertStatus(401);
    $this->get('/api/token')->assertStatus(401);
    $this->get('/api/token/my_token_name')->assertStatus(401);
    $this->put('/api/token/my_token_name')->assertStatus(401);
    $this->delete('/api/token/my_token_name')->assertStatus(401);
}

/**
 * Checking response codes during authentication.
 *
 * @return void
 */
public function test_authorized()
{
    $this->post('/api/login', [
        'nick' => env('TEST_USER_NICK'),
        'password' => env('TEST_USER_PASSWORD')
    ]->assertStatus(200);

    $this->get('/api/report/hours/short')->assertStatus(200);
    $this->get('/api/report/hours/project')->assertStatus(200);

    $tokenName = 'token ' . rand();
    $this->post('/api/token', [
        'name' => $tokenName
    ]->assertStatus(201);
}

```

```

$this->get('/api/token')->assertStatus(200);
$this->get("/api/token/$tokenName")->assertStatus(200);
$this->put("/api/token/$tokenName")->assertStatus(200);
$this->delete("/api/token/$tokenName")->assertStatus(204);
    }
}

```

tests/Feature/CRUDTokenTest.php

```
<?php
```

```
namespace Tests\Feature;
```

```
use Illuminate\Support\Facades\Validator;
```

```
use Tests\TestCase;
```

```
class CRUDTokenTest extends TestCase
```

```

{
    /**
     * {@inheritdoc}
     * @see \Illuminate\Foundation\Testing\TestCase::setUp()
     */
    protected function setUp(): void
    {
        parent::setUp();
        $this->authorization();
    }

    /**
     * Create
     *
     * @return void
     */
}

```

```

*/
public function test_create()
{
    $response = $this->get('/api/token/test token');
    $response->assertStatus(404);

    $response = $this->post('/api/token', ['name' => 'test token']);
    $response->assertStatus(201);
    $data = $response->json();

    $validator = Validator::make($data, [
        'data.name' => ['required', 'string'],
        'data.token' => ['required', 'string'],
    ]);

    $errors = $validator->errors()->all();
    $this->assertEmpty(
        $errors,
        "The application returned an incorrect response to the request. " .
        "Errors:\n" . implode("\n", $errors)
    );

    $response = $this->get('/api/token/test token');
    $response->assertStatus(200);
}

/**
 * Read
 *
 * @return void
 */

```

```

public function test_read()
{
    $response = $this->get('/api/token');
    $response->assertStatus(200);
    $data = $response->json();

    $validator = Validator::make($data, [
        'data.tokens' => ['required', 'array'],
        'data.tokens.*.name' => ['required', 'string'],
        'data.tokens.*.abilities' => ['required', 'array'],
        'data.tokens.*.abilities.*' => ['required', 'string'],
        'data.tokens.*.last_used_at' => ['date', 'nullable'],
    ]);

    $errors = $validator->errors()->all();
    $this->assertEmpty(
        $errors,
        "The application returned an incorrect response to the request. " .
        "Errors:\n" . implode("\n", $errors)
    );
}

/**
 * Read one
 *
 * @return void
 */
public function test_read_one()
{
    $response = $this->get('/api/token/test token');
    $response->assertStatus(200);

```

```

$data = $response->json();

$validator = Validator::make($data, [
    'data.name' => ['required', 'string'],
    'data.abilities' => ['required', 'array'],
    'data.abilities.*' => ['required', 'string'],
    'data.last_used_at' => ['date', 'nullable'],
]);

$errors = $validator->errors()->all();
$this->assertEmpty(
    $errors,
    "The application returned an incorrect response to the request. " .
    "Errors:\n" . implode("\n", $errors)
);
}

/**
 * Update
 *
 * @return void
 */
public function test_update()
{
    $response = $this->get('/api/token/test token');
    $response->assertStatus(200);
    $response = $this->get('/api/token/new name test token');
    $response->assertStatus(404);

    $response = $this->put('/api/token/test token', [
        'name' => 'new name test token'
    ]

```

```

    });
    $response->assertStatus(200);
    $data = $response->json();

    $validator = Validator::make($data, [
        'data.name' => ['required', 'string'],
        'data.abilities' => ['required', 'array'],
        'data.abilities.*' => ['required', 'string'],
        'data.last_used_at' => ['date', 'nullable'],
    ]);

    $errors = $validator->errors()->all();
    $this->assertEmpty(
        $errors,
        "The application returned an incorrect response to the request. " .
        "Errors:\n" . implode("\n", $errors)
    );
    $data = $validator->validated();
    $this->assertSame(
        'new name test token',
        $data['data']['name'],
        "Token {$data['data']['name']} did not accept the new name."
    );

    $response = $this->get('/api/token/test token');
    $response->assertStatus(404);
    $response = $this->get('/api/token/new name test token');
    $response->assertStatus(200);
}

/**

```



```

* Delete
*
* @return void
*/
public function test_delete()
{
    $response = $this->delete('/api/token/new name test token');
    $response->assertStatus(204);
    $response = $this->get('/api/token/new name test token');
    $response->assertStatus(404);
}
}

```

tests/Feature/LdapAuthTest.php

```
<?php
```

```
namespace Tests\Feature;
```

```
use Illuminate\Support\Facades\Auth;
```

```
use Tests\TestCase;
```

```
class LdapAuthTest extends TestCase
```

```

{
    public function test_auth_in_application()
    {
        $credentials = [
            'nick' => env('TEST_USER_NICK'),
            'password' => env('TEST_USER_PASSWORD'),
        ];
        $this->assertTrue(
            Auth::attempt($credentials),

```

```

        'Testuser authorization failed'
    );
}
}

```

tests/Feature/ModelsConnectionsTest.php

```
<?php
```

```
namespace Tests\Feature;
```

```
use App\Models\Project;
```

```
use App\Models\User;
```

```
use Tests\TestCase;
```

```
use App\Models\Vacation;
```

```
class ModelsConnectionsTest extends TestCase
```

```

{
    public function test_user()
    {
        foreach (User::all() as $user)
        {
            if(is_null($user->worker))
            {
                continue;
            }
            $this->assertTrue(
                $user->is($user->worker->user),
                "Model worker didn't related with User {$user->nick}."
            );
        }
    }
}

```

```

public function test_project()
{
    foreach (Project::all() as $project)
    {
        foreach ($project->blogRecords as $blogRecord)
        {
            $this->assertTrue(
                $project->is($blogRecord->project),
                "Model blogRecord({$blogRecord->id}) didn't related " .
                "with project({$project->id})."
            );
        }
    }
}

```

```

public function test_vacation()
{
    $numberRecords = 30;
    Vacation::truncate();
    Vacation::factory($numberRecords)->create();
    $records = Vacation::joinRelationship('worker')->get();
    $this->assertEquals(
        $numberRecords,
        count($records),
        "Join Vacation and Worker return uncorrect number records."
    );
}
}

```

tests/Feature/ReportHoursTest.php

&lt;?php

namespace Tests\Feature;

use App\Models\BlogRecord;

use App\Models\DailyBlog;

use App\Models\ReportHours;

use Illuminate\Support\Facades\Validator;

use Tests\TestCase;

class ReportHoursTest extends TestCase

{

/\*\*

\* Short report sum matching test to check that all records are retrieved

\* and everything is considered correct.

\*

\* @return void

\*/

public function test\_short\_report\_by\_simple\_sum()

{

\$report = new ReportHours();

\$totalSum = 0;

foreach (\$report-&gt;reportShort() as \$reportData)

{

\$totalSum += \$reportData['total\_hours'];

}

\$blogSum = BlogRecord::sum('time');

\$delta = abs(\$blogSum - \$totalSum);

\$this-&gt;assertEqualsWithDelta(

```
$blogSum,
```

```
$totalSum,
```

```
0.00001,
```

```
<<<MSG
```

```
The sum of hours of all blogs and the sum of hours of all reports
```

```
differ by more than the allowed error. The current difference is
```

```
$delta.
```

```
MSG
```

```
);
```

```
}
```

```
/**
```

```
* Report sum matching test to check that all records are retrieved
```

```
* and everything is considered correct.
```

```
*
```

```
* @return void
```

```
*/
```

```
public function test_report_by_project_by_simple_sum()
```

```
{
```

```
    $report = new ReportHours();
```

```
    $totalSum = 0;
```

```
    foreach ($report->reportByProject() as $reportData)
```

```
    {
```

```
        $totalSum += $reportData['total_hours'];
```

```
    }
```

```
    $blogSum = BlogRecord::sum('time');
```

```
    $delta = abs($blogSum - $totalSum);
```

```
    $this->assertEqualsWithDelta(
```

```
        $blogSum,
```

```
        $totalSum,
```

```
        0.00001,
```

```
<<<MSG
```

```
The sum of hours of all blogs and the sum of hours of all reports
    differ by more than the allowed error. The current difference is
```

```
$delta.
```

```
MSG
```

```
);
```

```
}
```

```
/**
```

```
* Check for correct filtering by date.
```

```
*
```

```
* @return void
```

```
*/
```

```
public function test_correct_time_filter()
```

```
{
```

```
    // Delete random values.
```

```
    BlogRecord::joinRelationship('dailyBlog')
```

```
        ->where(
```

```
            'date',
```

```
            '>=',
```

```
            date_create('2001-08-10')
```

```
        )
```

```
        ->where(
```

```
            'date',
```

```
            '<=',
```

```
            date_create('2016-08-10')
```

```
        )->delete();
```

```
    // Generation of known values.
```

```
    $day = DailyBlog::factory()->make();
```

```
    $day->date = date_create('@' .
```

```
        rand(
```

```

        997376400, // 2001-08-10
        1470762000, // 2016-08-10
    )
);
$day->save();
BlogRecord::factory(10)
    ->state([
        'blog_id' => $day->id,
        'time' => 2.1
    ])
    ->create();

$report = new ReportHours('10.08.2001', '10.08.2016');
$totalSum = 0;
foreach ($report->reportShort() as $reportData)
{
    $totalSum += $reportData['total_hours'];
}
$this->assertEqualsWithDelta(
    21,
    $totalSum,
    0.00001,
    <<<MSG
    The sum of hours of all blogs and the sum of hours of all reports
    differ by more than the allowed error. The current sum is $totalSum.
    MSG
);
}

public function test_short_report_http_request()
{

```

```

$this->authorization();

$response = $this->get('api/report/hours/short');
$response->assertStatus(200);
$data = $response->json();

$validator = Validator::make($data, [
    'data.count_reports' => ['required', 'integer'],
    'data.reports' => ['array'],
    'data.reports.*.nick' => ['required', 'string'],
    'data.reports.*.name' => ['required', 'string'],
    'data.reports.*.company' => ['required', 'string'],
    'data.reports.*.total_hours' => ['required', 'numeric'],
]);

$errors = $validator->errors()->all();
$this->assertEmpty(
    $errors,
    "The application returned an incorrect response to the request. " .
    "Errors:\n" . implode("\n", $errors)
);

$this->assertEquals(
    $data['data']['count_reports'],
    count($data['data']['reports']),
    'The count_reports field should show the correct number of reports.'
);
}

public function test_report_by_project_http_request()
{
    $this->authorization();

```



```

$response = $this->get('api/report/hours/project');
$response->assertStatus(200);
$data = $response->json();

```

```

$validator = Validator::make($data, [
    'data.count_reports' => ['required', 'integer'],
    'data.reports' => ['array'],
    'data.reports.*.nick' => ['required', 'string'],
    'data.reports.*.name' => ['required', 'string'],
    'data.reports.*.company' => ['required', 'string'],
    'data.reports.*.customer_id' => ['required', 'integer'],
    'data.reports.*.customer_name' => ['required', 'string'],
    'data.reports.*.project_id' => ['required', 'integer'],
    'data.reports.*.project_name' => ['required', 'string'],
    'data.reports.*.total_hours' => ['required', 'numeric'],
]);

```

```

$errors = $validator->errors()->all();
$this->assertEmpty(
    $errors,
    "The application returned an incorrect response to the request. " .
    "Errors:\n" . implode("\n", $errors)
);

```

```

$this->assertEquals(
    $data['data']['count_reports'],
    count($data['data']['reports']),
    'The count_reports field should show the correct number of reports.'
);

```

```

}

```

```

}

```

tests/Feature/SheduleTest.php

```

<?php

namespace Tests\Feature;

use Illuminate\Support\Facades\Validator;
use Tests\TestCase;

class SheduleTest extends TestCase
{
    public function setUp(): void
    {
        parent::setUp();
        $this->authorization();
    }

    /**
     * Correct request test.
     *
     * @return void
     */
    public function test_shedule_official_validate()
    {
        $response = $this->get('api/shedule/official?start_date=2010-08-
15&end_date=2010-08-15');
        $response->assertStatus(200);
        $data = $response->json();

        $validator = Validator::make($data, [
            'data.days' => ['required', 'array'],

```

```

        'data.days.*.status' => ['required', 'string'],
        'data.days.*.info' => ['array'],
        'data.days.*.info.*' => ['required', 'string'],
    ]);

    $errors = $validator->errors()->all();
    $this->assertEmpty(
        $errors,
        "The application returned an incorrect response to the request. " .
        "Errors:\n" . implode("\n", $errors)
    );
}

/**
 * Correct request test.
 *
 * @return void
 */
public function test_shedule_workers_validate()
{
    $response = $this->get('api/shedule/worker?start_date=2010-08-
15&end_date=2010-08-15');
    $response->assertStatus(200);
    $data = $response->json();

    $validator = Validator::make($data, [
        'data.*' => ['required', 'array'],
        'data.*.days.*.status' => ['required', 'string'],
        'data.*.days.*.total_hours' => ['number'],
        'data.*.days.*.info' => ['array'],
        'data.*.days.*.info.*' => ['required', 'string'],
    ]);
}

```

```

    });

    $errors = $validator->errors()->all();
    $this->assertEmpty(
        $errors,
        "The application returned an incorrect response to the request. " .
        "Errors:\n" . implode("\n", $errors)
    );
}
}

```

tests/TestCase.php

```

<?php

namespace Tests;

use App\Models\User;
use Illuminate\Foundation\Testing\TestCase as BaseTestCase;
use Laravel\Sanctum\Sanctum;

abstract class TestCase extends BaseTestCase
{
    use CreatesApplication;

    protected function authorization()
    {
        Sanctum::actingAs(
            User::firstWhere('nick', env('TEST_USER_NICK'))
        );
    }
}

```