install:api
laravel new adibidea04
    No starter kit
php artisan install:api
The install:api command installs Laravel Sanctum, which provides a robust, yet simple API token authentication guard which can be used to authenticate third-party API consumers, SPAs, or mobile applications. In addition, the install:api command creates the routes/api.php file

make:model -a --api
php artisan make:model Post -a --api

  INFO Model [app/Models/Post.php] created successfully.
  INFO Factory [database/factories/PostFactory.php] created successfully.
  INFO Migration [database/migrations/2024_12_10_161437_create_posts_table.php] created successfully.
  INFO Seeder [database/seeders/PostSeeder.php] created successfully.
  INFO Request [app/Http/Requests/StorePostRequest.php] created successfully.
  INFO Request [app/Http/Requests/UpdatePostRequest.php] created successfully.
  INFO Controller [app/Http/Controllers/PostController.php] created successfully.
  INFO Policy [app/Policies/PostPolicy.php] created successfully.
app/Modesls/Post.php:

```
<?php

...
class Post extends Model
{
    /** @use HasFactory<\Database\Factories\PostFactory> */
    use HasFactory;

    protected $fillable = [
        'title',
        'body'
    ];

...
}
```
Posts migration:

```
...
        Schema::create('posts', function (Blueprint $table) {
            $table->id();
            $table->string('title');
            $table->text('body');
            $table->timestamps();
        });
```
 php artisan migrate:refresh

routes/api.php:

```
use App\Http\Controllers\PostController;

Route::apiResource('posts', PostController::class);
```

```
php artisan route:list --name=posts

  GET|HEAD        api/posts ..................................................... posts.index › PostController@index
  POST            api/posts ..................................................... posts.store › PostController@store
  GET|HEAD        api/posts/{post} ............................................... posts.show › PostController@show
  PUT|PATCH       api/posts/{post} ............................................... posts.update › PostController@update
  DELETE          api/posts/{post} ............................................... posts.destroy › PostController@destroy
```

Post controller:

```
<?php

namespace App\Http\Controllers;

use App\Models\Post;
use Illuminate\Http\Request;

class PostController extends Controller
{
    /**
     * Display a listing of the resource.
     */
    public function index()
    {
        //return Post::all();
        return Post::with('user')->latest()->get();
    }

    /**
     * Store a newly created resource in storage.
     */
    public function store(Request $request)
    {
        $fields = $request->validate([
            'title' => 'required:max:255',
            'body' => 'required'
        ]);

        $post = Post::create($fields);
        //return ['post' => $post];
```

```
        return ['post' => $post, 'user' => $post->user];
    }

    /**
     * Display the specified resource.
     */
    public function show(Post $post)
    {
        //return $post;
        return ['post' => $post, 'user' => $post->user];
    }

    /**
     * Update the specified resource in storage.
     */
    public function update(Request $request, Post $post)
    {
        $fields = $request->validate([
            'title' => 'required:max:255',
            'body' => 'required'
        ]);

        $post->update($fields);

        //return $post;
        return ['post' => $post, 'user' => $post->user];
    }

    /**
     * Remove the specified resource from storage.
     */
    public function destroy(Post $post)
    {
        $post->delete();

        return ['message' => 'The post was deleted'];
    }
}
```

Rest client request

request.http fitxategia sortu eta Rest Client extension instalatu

```
@url = http://localhost:8000

### GET posta guztiak
GET {{url}}/api/posts

### POST posta create post bat
POST {{url}}/api/posts
```

```
Accept: application/json
Content-Type: application/json

{
   "title" : "Titulua 1",
   "body" : "Edukia 1"
}

### GET posta id=1 duena
GET {{url}}/api/posts/1

### PUT posta eguneratu
PUT {{url}}/api/posts/2
Accept: application/json
Content-Type: application/json

{
   "title" : "Titulua 2 eguneratua",
   "body" : "Edukia 2 eguneratua"
}

### DELETE posta bat
DELETE {{url}}/api/posts/1
API Authentication token. Login
laravel

HasApiTokens gehitu app/Models/User.php:
(https://laravel.com/docs/11.x/sanctum#issuing-api-tokens)

use Laravel\Sanctum\HasApiTokens;

class User extends Authenticatable
{
   /** @use HasFactory<\Database\Factories\UserFactory> */
   use HasFactory, Notifiable, HasApiTokens;
...
php artisan make:controller AuthController
app/Http/Controllers/AuthController.php:

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Models\User;
use Illuminate\Support\Facades\Hash;

class AuthController extends Controller
```

```
{
    public function register(Request $request) {
        // https://laravel.com/docs/11.x/validation
        $fields = $request->validate([
            'name' => 'required|max:255',
            'email' => 'required|email|unique:users',
            'password' => 'required|confirmed'
        ]);

        $user = User::create($fields);

        $token = $user->createToken($request->name);

        return [
            'user' => $user,
            'token' => $token->plainTextToken
        ];

        // return [
        //     'user' => $user,
        //     'token' => $token
        // ];
    }

    public function login(Request $request) {

        $fields = $request->validate([
            'email' => 'required|email|exists:users',
            'password' => 'required'
        ]);

        $user = User::where('email', $request->email)->first();

        if (!$user || !Hash::check($request->password, $user->password)) {
            return response(['message' => 'The provided credentials are incorrect.'],  401);

            //return [
            //    'message' => 'The provided credentials are incorrect.'
            //];
        }

        $token = $user->createToken($user->name);

        return [
            'user' => $user,
            'token' => $token->plainTextToken
        ];
```

```
    }

    public function logout(Request $request) {

        # tokenak ezabatu logout egitean
        # https://laravel.com/docs/11.x/sanctum#revoking-tokens
        $request->user()->tokens()->delete();

        return [
            'message' => 'You are logout.'
        ];
    }

}
```

routes/api.php

```
use App\Http\Controllers\AuthController;

Route::post('/register', [AuthController::class, 'register']);
Route::post('/login', [AuthController::class, 'login']);
```

user registratu:

```
### POST user register
POST {{url}}/api/register
Accept: application/json
Content-Type: application/json

{
    "name" : "Koxme",
    "email" : "koxme@koxme.koxme",
    "password" : "pasahitza",
    "password_confirmation" : "pasahitza"
}
```

erantzuna:

```
{
  "user": {
    "name": "Koxme",
    "email": "koxme@koxme.koxme",
    "updated_at": "2024-12-10T18:41:37.000000Z",
    "created_at": "2024-12-10T18:41:37.000000Z",
    "id": 1
  },
  "token": "1|yIg7mC1Oo7OsNW2KhdOeEBmOyuAP80z4vwnPLnP6c44d0d72"
}
```

login:

```
### POST user login
```

```
# @name login
POST {{url}}/api/login
Accept: application/json
Content-Type: application/json

{
   "email" : "koxme@koxme.koxme",
   "password" : "pasahitza"
}
```

logout egiteko autentikatua egon behar da (# https://laravel.com/docs/11.x/sanctum#api-token-authentication)

routes/api.php:

```
Route::post('/logout', [AuthController::class, 'logout'])->middleware('auth:sanctum');
```

Bearer token erabili behar da, token-a login egitean lortzen da:

```
### POST user logout
POST {{url}}/api/logout
Authorization: Bearer {{token}}
#Accept: application/json
```

[User]<--1:N-->[Post]

Post-ak babestuko ditugu. Post-ak user-ak lotuko ditugu 1:N erlazioarekin eta gero babestu:

[User]<--1:N-->[Post]

Post.php

```
class Post extends Model
{
...
   public function user() {
      return $this->belongsTo(User::class);
   }
```

User.php

```
class User extends Authenticatable
{
...
   public function posts() {
      return $this->hasMany(Post::class);
   }
```

Post migrazioan:

```
            # https://laravel.com/docs/11.x/migrations#foreign-key-constraints
            $table->foreignId('user_id')->constrained()->cascadeOnDelete();
```

Orain post berri bat sortzean, logina egin duen user-ari gehituko zaio:

PostController.php ():

```
        $post = $request->user()->posts()->create($fields);
        return $post;
```

php artisan migrate:fresh
Orain post-ak babestuko ditugu middleware bidez:

PostController.php:

```
use Illuminate\Routing\Controllers\HasMiddleware;
use Illuminate\Routing\Controllers\Middleware;

class PostController extends Controller implements HasMiddleware
{

    public static function middleware()
    {
        return [
            new middleware('auth:sanctum', except: ['index', 'show'])
        ];
    }
```

Hemendik aurrera, autentikazioa eskatzen duten http request guztietan Authorization: Bearer {{token}} jarri behar da:

```
### POST posta create post bat
POST {{url}}/api/posts
Authorization: Bearer {{token}}
Accept: application/json
Content-Type: application/json

{
    "title" : "Titulua 2",
    "body" : "Edukia 2"
}
```

erabiltzaile batek beste erabiltzaile baten post-ak ezabatu ditzake.

app/policies/PostPolicy.php:

```
<?php

namespace App\Policies;

use App\Models\Post;
use App\Models\User;
use Illuminate\Auth\Access\Response;

class PostPolicy
{
```

```
    public function modify(User $user, Post $post): Response
    {
        return $user->id === $post->user_id
            ? Response::allow()
            : Response::deny('You do not own this post');
    }
}
```

PostController.php

```
use Illuminate\Support\Facades\Gate;

    public function update(Request $request, Post $post)
    {
        Gate::authorize('modify', $post);
...
    public function destroy(Post $post)
    {
        Gate::authorize('modify', $post);
...
```

token expiration:

config/sanctum.php

```
    /*
    |--------------------------------------------------------------------------
    | Expiration Minutes
    |--------------------------------------------------------------------------
    |
    | This value controls the number of minutes until an issued token will be
    | considered expired. This will override any values set in the token's
    | "expires_at" attribute, but first-party sessions are not affected.
    |
    */

    'expiration' => null,
```

Factory Seed

posts-ak automatikoki datuz betetzeko.

beharrezko fitxategian sortuak ditugu modeloak sortzean -a aukera erabili delako.

PostFactory.php

```
<?php

namespace Database\Factories;

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

```
/**
 * @extends \Illuminate\Database\Eloquent\Factories\Factory<\App\Models\Post>
 */
class PostFactory extends Factory
{
    protected $model = \App\Models\Post::class;

    public function definition(): array
    {
        return [
            'title' => $this->faker->sentence,
            'body' => $this->faker->paragraph(5),
            'user_id' => 1  // begiratu DatabaseSeeder, bertan "test user" id=1 duena sortzen da
        ];
    }
}
```

PostSeeder.php:

```
<?php

namespace Database\Seeders;

use Illuminate\Database\Console\Seeds\WithoutModelEvents;
use Illuminate\Database\Seeder;
use App\Models\Post;

class PostSeeder extends Seeder
{
    /**
     * Run the database seeds.
     */
    public function run(): void
    {
        // Create 10 posts
        Post::factory(10)->create();
    }
}
```

DatabaseSeeder.php:

```
<?php

namespace Database\Seeders;

use App\Models\User;
// use Illuminate\Database\Console\Seeds\WithoutModelEvents;
use Illuminate\Database\Seeder;

class DatabaseSeeder extends Seeder
```

```
{
    /**
     * Seed the application's database.
     */
    public function run(): void
    {
        // User::factory(10)->create();

        User::factory()->create([
            'name' => 'Koxme',
            'email' => 'koxme@koxme.koxme',
            'password' => 'pasahitza',
        ]);

        // posts
        $this->call([
            PostSeeder::class,
        ]);
    }
}
php artisan migrate:fresh --seed
# edo dena berregiteko
php artisan migrate:refresh --seed
```