**1. Структура проекта Laravel**

Copy

laravel-project/

├── app/

│ ├── Http/

│ │ ├── Controllers/

│ │ │ ├── CustomerController.php

│ │ │ ├── OrderController.php

│ │ │ └── ... другие контроллеры

│ ├── Models/

│ │ ├── Customer.php

│ │ ├── Order.php

│ │ └── ... другие модели

├── database/

│ ├── migrations/

│ │ ├── 2023\_01\_01\_000000\_create\_customers\_table.php

│ │ ├── 2023\_01\_01\_000001\_create\_orders\_table.php

│ │ └── ... другие миграции

│ ├── seeders/

├── routes/

│ ├── api.php

│ └── web.php

└── resources/

└── views/

**2. Пошаговая инструкция по размещению кода**

**2.1. Миграции**

Размещаются в database/migrations/

Как создать:

bash

Copy

php artisan make:migration create\_customers\_table

php artisan make:migration create\_orders\_table

# и т.д. для каждой таблицы

Вставьте код каждой миграции в соответствующий файл, например:

php

Copy

// database/migrations/2023\_01\_01\_000000\_create\_customers\_table.php

use Illuminate\Database\Migrations\Migration;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Support\Facades\Schema;

return new class extends Migration

{

public function up()

{

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

$table->id();

$table->string('name');

// ... остальные поля

$table->timestamps();

});

}

};

**2.2. Модели**

Размещаются в app/Models/

Как создать:

bash

Copy

php artisan make:model Customer

php artisan make:model Order

# и т.д. для каждой модели

Пример для модели Customer:

php

Copy

// app/Models/Customer.php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class Customer extends Model

{

protected $fillable = ['name', 'email', 'phone'];

public function orders()

{

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

}

}

**2.3. Контроллеры**

Размещаются в app/Http/Controllers/

Как создать:

bash

Copy

php artisan make:controller CustomerController

php artisan make:controller OrderController

Пример для OrderController:

php

Copy

// app/Http/Controllers/OrderController.php

namespace App\Http\Controllers;

use App\Models\Order;

use Illuminate\Http\Request;

class OrderController extends Controller

{

public function store(Request $request)

{

// ... ваш код

}

}

**2.4. Маршруты (роуты)**

Размещаются в routes/api.php (для API) или routes/web.php (для веб-интерфейса)

Пример для api.php:

php

Copy

// routes/api.php

use App\Http\Controllers\OrderController;

Route::apiResource('orders', OrderController::class);

Route::post('orders/{order}/payments', [OrderController::class, 'addPayment']);

**2.5. Калькулирующие запросы**

Обычно размещаются:

1. В моделях (как методы)
2. В репозиториях (если используется этот паттерн)
3. Непосредственно в контроллерах

Пример в модели Order:

php

Copy

// app/Models/Order.php

public function calculateTotal()

{

return $this->items->sum(function($item) {

return $item->quantity \* $item->unit\_price;

});

}

**2.6. Запуск миграций**

После создания всех миграций выполните:

bash

Copy

php artisan migrate

**2.7. Сидеры (начальные данные)**

Создайте сидеры для тестовых данных:

bash

Copy

php artisan make:seeder CustomersTableSeeder

Заполните его:

php

Copy

// database/seeders/CustomersTableSeeder.php

public function run()

{

\App\Models\Customer::factory(10)->create();

}

Запустите сидеры:

bash

Copy

php artisan db:seed

**3. Порядок реализации**

1. Сначала создайте миграции для всех таблиц
2. Затем создайте модели с отношениями
3. Добавьте контроллеры с базовыми методами (index, store, show и т.д.)
4. Настройте маршруты в api.php или web.php
5. Реализуйте бизнес-логику (калькуляции, проверки)
6. Добавьте сидеры для тестовых данных
7. Протестируйте API через Postman или фронтенд

**4. Дополнительные советы**

1. Для сложных запросов создайте отдельные классы в app/Services/
2. Для валидации создайте FormRequest классы:

bash

Copy

php artisan make:request StoreOrderRequest

1. Для тестирования создайте тесты:

bash

Copy

php artisan make:test OrderTest

Все компоненты должны работать вместе, образуя законченное приложение с:

* Моделями (бизнес-логика)
* Контроллерами (обработка запросов)
* Маршрутами (доступ к функционалу)
* Миграциями (структура БД)