REPOSITORY PATTERN LARAVEL UYGULAMASI

Dosya Yapısı:

```
|_app
    |_Repository
        |_Eloquent
        -BaseRepository.php
        -UserRepository.php
        -ProductRepository.php
            |_Contracts
            -EloquentRepositoryInterface.php
            -UserRepositoryInterface.php
            -ProductRepositoryInterface.php
```

EloquentRepositoryInterface:
Ortak kullanılacak metotlar için tanımlamalar içerir.

```
namespace App\Repository\Eloquent\Contracts;

use Illuminate\Database\Eloquent\Model;
use Illuminate\Support\Collection;

interface EloquentRepositoryInterface {
   public function getAll(): Collection;
   public function getById(int $id): ?Model;
   public function create(array $collection=[]): Model;
   public function update(int $id, array $collection=[]): bool;
   public function delete(int $id): bool;
}
```

BaseRepository:
Ortak kullanılacak metotların implementasyonlarını içerir.

```
namespace App\Repository\Eloquent;

use App\Repository\Eloquent\Contracts\EloquentRepositoryInterface;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Support\Collection;

class BaseRepository implements EloquentRepositoryInterface {
   protected $model;

   public function __construct(Model $model) {
      $this->model = $model;
   }
   public function getAll(): Collection {
      return $this->model->all();
   }
   public function getById(int $id): ?Model {
      return $this->model->find($id);
   }
   public function create(array $collection=[]): Model {
```

```
        return $this->model->create($collection);
    }
    public function update(int $id, array $collection=[]): bool {
        $this->model = $this->getById($id);
        return $this->model->update($collection);
    }
    public function delete(int $id): bool {
        return $this->model->destroy($id);
    }
}
```

UserRepositoryInterface:
Kullanıcılara özel veritabanı erişim metot tanımlamalarını içerir.

```
namespace App\Repository\Eloquent\Contracts;

use App\Models\User;

interface UserRepositoryInterface {
        public function getByProductId(int $id): ?User;
}
```

UserRepository:
Kullanıcılara özel veritabanı erişim metot implementasyonlarını içerir.

```
namespace App\Repository\Eloquent;

use App\Models\User;
use App\Repository\Eloquent\Contracts\UserRepositoryInterface;

class UserRepository extends BaseRepository implements UserRepositoryInterface {
    public function __construct(User $model) {
        parent::__construct($model);
    }
    public function getByProductId(int $id): ?User {
        return $this->model->where("product_id", $id)->get();
    }
}
```

ProductRepositoryInterface:
Ürünlere özel veritabanı erişim metot tanımlamalarını içerir.

```
namespace App\Repository\Eloquent\Contracts;

use App\Models\Product;

interface ProductRepositoryInterface {
        public function getByUserId(int $id): ?Product;
}
```

ProductRepository:
Ürünlere özel veritabanı erişim metot implementasyonlarını içerir.

```
namespace App\Repository\Eloquent;

use App\Models\Product;
use App\Repository\Eloquent\Contracts\ProductRepositoryInterface;

class ProductRepository extends BaseRepository implements ProductRepositoryInterface {
   public function __construct( Product $model) {
      parent::__construct($model);
   }
   public function getByUserId(int $id): ?Product {
      return $this->model->where("user_id", $id)->get();
   }
}
```

Laravel service container içine dahil etmek için service provider oluşturulması gerekmektedir.

```
php artisan make:provider RepositoryServiceProvider
```

RepositoryServiceProvider:
Service provider register metodunda, veritabanı nesnelerinin tekillik oluşturması amacıyla sınıf ve arayüzler singleton ile bind edilmelidir.

```
namespace App\Providers;

use App\Repository\Eloquent\Contracts\EloquentRepositoryInterface;
use App\Repository\Eloquent\Contracts\UserRepositoryInterface;
use App\Repository\Eloquent\Contracts\ProductRepositoryInterface;
use App\Repository\Eloquent\UserRepository;
use App\Repository\Eloquent\ProductRepository;
use App\Repository\Eloquent\BaseRepository;
use Illuminate\Support\ServiceProvider;

class RepositoryServiceProvider extends ServiceProvider
{
  public function register()
  {
     $this->app->singleton(EloquentRepositoryInterface::class, BaseRepository::class);
     $this->app->singleton(UserRepositoryInterface::class, UserRepository::class);
     $this->app->singleton(ProductRepositoryInterface::class, ProductRepository::class);
  }
}
```

UserController:
Repository sınıfı metotları controller içinde hemen hemen tüm metotlarda kullanılacaksa constructor aracılığıyla inject edilebilir. Eğer tek bir metot veya birkaç metotta kullanılacaksa metot tanımında inject edilmesi faydalıdır.

```
namespace App\Http\Controllers;

use App\Repository\Eloquent\Contracts\UserRepositoryInterface;
use App\Repository\Eloquent\Contracts\ProductRepositoryInterface;

class UsersController extends Controller
{
  private $userRepository;

  public function __construct(UserRepositoryInterface $userRepository)
  {
     $this->userRepository = $userRepository;
  }
  public function index()
  {
     $users = $this->userRepository->all();

     return view('users.index', [
        'users' => $users
     ]);
  }
  public function show()
  {
     $user_id = Auth::user()->id;
     $user = $this->userRepository->getById($user_id);

     return view('users.show', [
        'user' => $user
     ]);
  }
  public function create()
  {
      $collection = [
         "name" => $request->name,
         "email" => $request->email
      ];
     $user = $this->userRepository->create($collection);
  }
  public function list($product_id)
  {
     $user = $this->userRepository->getByProductId($product_id);

     return view('users.list', [
        'users' => $users
     ]);
  }
  public function productList(ProductRepositoryInterface $productRepository)
```

```
    {
        $user_id = Auth::user()->id;
        $products = $productRepository->getByUserId($user_id);

        return view('users.product_list', [
            'products' => $products
        ]);
    }
}
```

Referanslar:

https://asperbrothers.com/blog/implement-repository-pattern-in-laravel/