Web Programlama

Laravel Fremework ile geliştirilmiş Web sitesi

Zeynep Hepgüler *(Bilişim Sistemleri Mühendisliği)* KOCAELİ/Derince
Kocaeli Üniversitesi 201307080 zeynephepguler@gmail.com

I. GIRIŞ

Laravel, PHP programlama dilinde geliştirilmiş bir web uygulama çatısıdır. Bu raporda, Laravel ile ilgili temel kavramlar ve işlemler hakkında bilgi vereceğiz. Bunlar arasında MySQL veritabanı bağlama işlemleri, Laravel içinden tablo oluşturma işlemleri, Laravel'in Controller sınıfını kullanma işlemleri, Laravel'in Migration sınıfını kullanma işlemleri ve Laravel'in Model dosyalarını kullanma işlemleri yer almaktadır.

II. MYSQL VERITABANI BAĞLAMA İŞLEMLERI

Laravel, MySQL veritabanına bağlanmak için kullanabileceğiniz birçok yol sunar. Bunlar arasında PHP PDO ve MySQLi sınıfları yer almaktadır. Bununla birlikte, Laravel, veritabanı bağlantısını kolaylaştırmak için bir ORM (Object-Relational Mapping) katmanı sağlar. Bu ORM katmanı, veritabanı işlemlerini PHP sınıfları aracılığıyla gerçekleştirir.

Veritabanına bağlanmak için Laravel'in konfigürasyon dosyasında veritabanı bilgileri tanımlanır. Bu bilgiler, host adı, kullanıcı adı, şifre ve veritabanı adını içerir. Laravel, veritabanı bağlantısını sağlamak için bu bilgileri kullanır.

III. LARAVEL İÇINDEN TABLO OLUŞTURMA İŞLEMLERI

Laravel, veritabanı işlemlerini gerçekleştirmek için Migration sınıfını kullanır. Migration sınıfı, veritabanı tablolarını ve sütunlarını oluşturmak, güncellemek veya silmek için kullanılır. Migration sınıfı, PHP sınıfları aracılığıyla tanımlanır.

Migration sınıfı, tabloların ve sütunların oluşturulması için kolay bir yol sağlar. Laravel, Migration sınıfını kullanarak, veritabanı işlemlerini kolayca gerçekleştirebilirsiniz.

IV. LARAVEL'IN CONTROLLER SINIFINI KULLANMA İŞLEMLERI

Laravel, Controller sınıfını kullanarak, HTTP isteklerini işleyebilirsiniz. Controller sınıfı, HTTP isteklerini yönetmek için kullanılır. Bu sınıf, web uygulamasının işlevselliğini yönetir.

Controller sınıfı, web uygulamasının işlevselliğini yönetmek için kullanılır. Bu sınıf, web uygulamasının işlevselliğini yönetir.

V. LARAVEL'IN MIGRATION SINIFINI KULLANMA İŞLEMLERI

Laravel, Migration sınıfını kullanarak, veritabanı tablolarını ve sütunlarını oluşturabilir, güncelleyebilir veya silebilirsiniz. Migration sınıfı, PHP sınıfları aracılığıyla tanımlanır. Migration sınıfı, tabloların ve sütunların oluşturulması için kolay bir yol sağlar.

VI. LARAVEL'IN MODEL DOSYALARINI KULLANMA İŞLEMLERI

Laravel, Model sınıfını kullanarak, veritabanı tabloları ile etkileşimde bulunabilirsiniz. Model sınıfı, veritabanı tablolarını temsil eden PHP sınıflarıdır. Bu sınıflar, veritabanı işlemlerini gerçekleştirmek için kullanılır.

Model sınıfları, veritabanı tabloları ile etkileşimde bulunmak için birçok yöntem sağlar. Bu yöntemler arasında verileri sorgulama, verileri güncelleme, verileri silme ve verileri oluşturma yer alır. Model sınıfları, veritabanı işlemlerini kolaylaştırır ve kod tekrarını önler.

VII. HTML SAYFALARININ DÜZENI

Laravel, Blade şablon motorunu kullanarak, HTML sayfalarının düzenini yönetir. Blade, PHP tabanlı bir şablon motorudur. Bu şablon motoru, web uygulamasının görüntüsünü yönetmek için kullanılır.

Blade şablon motoru, web uygulamasının görüntüsünü yönetmek için birçok işlev sağlar. Bu işlevler arasında şablon kalıtımı, şablon bölümleri, şablon döngüleri ve şablon koşulları yer alır. Blade, HTML sayfalarının düzenini yönetmek için kullanılabilir.

Controller Sınıfında Yazılan Fonksiyonlar ve Web.php Dosyasına Route Ayarlama

Laravel, HTTP isteklerini Controller sınıfı aracılığıyla işler. Controller sınıfı, HTTP isteklerini işleyen PHP sınıflarıdır. Bu sınıflar, web uygulamasının işlevselliğini yönetir.

Web.php dosyası, HTTP isteklerini Controller sınıfına yönlendirmek için kullanılır. Bu dosya, web uygulamasının yönlendirmesini kontrol eder. Web.php dosyası, HTTP istekleri ile Controller sınıfındaki işlevler arasında bir köprü görevi görür.

Controller sınıfında yazılan fonksiyonlar, web.php dosyasına yönlendirmek için belirli bir formatta yazılır. Bu formatta, HTTP isteği türü, URL ve Controller sınıfındaki işlev adı belirtilir.

Örnek olarak, bir GET isteği için Controller sınıfında yazılan fonksiyonun web.php dosyasına yönlendirilmesi aşağıdaki şekilde olabilir:

```
Route::get('/urunler', 'UrunController@index');
```

Bu kod parçası, '/urunler' URL'sine yapılan GET isteğinin 'UrunController' sınıfındaki 'index' işlevine yönlendirilmesini sağlar.

VIII. VERITABANI BAĞLANTISI

A. *Env Dosya Düzeni*

Laravel'de .env dosyası, uygulamanın çeşitli konfigürasyon ayarlarının tutulduğu yerdir. Bu dosya, özellikle veritabanı bağlantısı gibi hassas bilgilerin depolanması için kullanılır. MySQL veritabanı bağlantısının oluşturulması için .env dosyasının nasıl yapılandırılacağını aşağıdaki adımlarla inceleyebilirsiniz:

1. MySQL veritabanı oluşturun: MySQL veritabanınız yoksa, bir veritabanı oluşturmanız gerekmektedir. Bu işlemi MySQL istemcisi, phpMyAdmin veya herhangi bir veritabanı yönetim aracı ile gerçekleştirebilirsiniz.

2. .env Dosyasını Açın: Uygulama klasöründe bulunan .env dosyasını bir metin düzenleyicide açın.

3. Veritabanı Bağlantısı Ayarlarını Yapılandırın: Aşağıdaki değişkenleri .env dosyasında yapılandırmanız gerekmektedir:
- DB_HOST: MySQL sunucusunun IP adresi veya alan adı
- DB_PORT: MySQL sunucusunun port numarası (varsayılan olarak 3306)
- DB_DATABASE: Veritabanının adı
- DB_USERNAME: Veritabanı kullanıcısı adı
- DB_PASSWORD: Veritabanı kullanıcısı şifresi

Örnek olarak:

```
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=mydatabase
DB_USERNAME=myusername
DB_PASSWORD=mypassword
```

4. .env Dosyasını Kaydedin: Yapılandırmanızı tamamladıktan sonra, .env dosyasını kaydedin.

5. Veritabanı Bağlantısını Yapılandırın: Veritabanı bağlantısını yapılandırmak için, config/database.php dosyasını açın. Burada MySQL veritabanı bağlantısını yapılandırmak için gerekli olan ayarları görebilirsiniz. Bu dosyada yapılandırmanız gereken tek şey, varsayılan olarak kullanılacak olan veritabanı bağlantısıdır.

Örnek olarak:

```
'default' => env('DB_CONNECTION', 'mysql'),
```

6. Veritabanı Bağlantısını Test Edin: Veritabanı bağlantısını test etmek için, terminal veya komut istemcisini açın ve aşağıdaki komutu çalıştırın:

```
php artisan migrate
```

Bu komut, veritabanınızda bir "migrations" tablosu oluşturacaktır. Bu işlem başarılı olursa, veritabanı bağlantısı yapılandırması doğru bir şekilde yapılmış demektir.

IX. OLUŞTURULAN TABLOLAR

Laravel'de tablo oluşturmak için şu adımları izleyebilirsiniz:

1. Öncelikle, oluşturacağınız tablonun modelini oluşturmanız gerekiyor. Bunun için terminalden `php artisan make:model TabloAdi` komutunu kullanabilirsiniz. Bu komut, `app/Models` dizini altında `TabloAdi.php` adlı bir dosya oluşturur.

2. Ardından, oluşturacağınız tablonun migration dosyasını oluşturmanız gerekiyor. Migration dosyası, veritabanında bir tablo oluşturmak için kullanılan bir PHP dosyasıdır. Migration dosyasını oluşturmak için terminalden `php artisan make:migration create_tablo_adı_table` komutunu kullanabilirsiniz. Bu komut, `database/migrations` dizini altında `create_tablo_adı_table.php` adlı bir dosya oluşturur.

3. Migration dosyasını açarak, `up()` fonksiyonunda tablonun sütunlarını tanımlayabilirsiniz. Örneğin, `string` tipinde bir `name` sütunu ve `integer` tipinde bir `age` sütunu oluşturmak isterseniz, `up()` fonksiyonunu şu şekilde yazabilirsiniz:

```
public function up()
{
    Schema::create('tablo_adi', function (Blueprint $table) {
        $table->id();
        $table->string('name');
        $table->integer('age');
        $table->timestamps();
    });
}
```

4. Migration dosyasını kaydettikten sonra, terminalde `php artisan migrate` komutunu kullanarak migration işlemini gerçekleştirebilirsiniz. Bu komut, veritabanında `tablo_adi` adlı bir tablo oluşturur.

5. Tabloyu veritabanından silmek için `php artisan migrate:rollback` komutunu kullanabilirsiniz. Bu komut, son yaptığınız migration işlemini geri alır.

Bu şekilde, Laravel'de tablo oluşturma işlemini gerçekleştirebilirsiniz.

A. Tablolar

1) User Tablosu

Bu özel migrasyon dosyası, `kullanicis` adında bir tablo oluşturur ve bu tablonun sütunlarını belirler. Bu sütunlar aşağıdaki gibidir:

- `id`: Otomatik artan bir tamsayı (primary key)
- `kullaniciAdi`: Null değer alabilen bir karakter dizisi sütunu
- `Soyad`: Benzersiz bir karakter dizisi sütunu
- `Telefon`: Null değer alabilen bir karakter dizisi sütunu
- `Adres`: Null değer alabilen bir karakter dizisi sütunu
- `Sifre`: Null değer alabilen bir karakter dizisi sütunu
- `Email`: Null değer alabilen bir karakter dizisi sütunu
- `created_at`: Oluşturma tarihi (timestamp)
- `updated_at`: Güncelleme tarihi (timestamp)

Bu dosyanın içindeki `up()` fonksiyonu, tabloyu oluşturmak için çalışır. `down()` fonksiyonu ise tablonun geri alınması için çalışır. Bu dosya, `php artisan migrate` komutu ile çalıştırıldığında veritabanına tablo ekleyecektir.

2) Urunler Tablosu

Bu özel migration dosyası, "urunlers" adlı bir tablo oluşturur. Tablonun sütunları şunlardır:
- id: tablodaki her bir satır için benzersiz bir kimlik numarası (primary key)
- gorsel: ürün görselini içeren bir sütun
- urunadi: ürün adını içeren bir sütun
- marka: ürün markasını içeren bir sütun
- model: ürün modelini içeren bir sütun
- bilgi: ürün hakkında açıklama veya bilgi içeren bir sütun
- fiyat: ürünün fiyatını içeren bir sütun
- kategori: ürünün kategorisini içeren bir sütun
- satici: ürünün satıcısını içeren bir sütun
- timestamps: Laravel'in oluşturma ve güncelleme tarihlerini otomatik olarak yönetmesi için gerekli olan iki sütun olan "created_at" ve "updated_at"

Terminalde "php artisan migrate" komutunu çalıştırdığınızda, bu migration dosyası yürütülerek, veritabanınızdaki ilgili tablo oluşturulacaktır.

3) Sepet Tablosu

Bu kod, Laravel migrasyon sınıfını kullanarak bir `sepets` tablosu oluşturmak için yazılmış bir migrasyon dosyasıdır. Aşağıdaki işlemleri gerçekleştirir:

1. `up()` fonksiyonunda, `sepets` adında bir tablo oluşturmak için `Schema::create()` metodu kullanılır. Tablonun sütunları belirlenir:

- `id`: tablo için birincil anahtar.
- `sepet_id`: unsignedBigInteger türünde bir sütun, sepetteki ürünlerin ilişkilendirilmesi için kullanılır.
- `urun_id`: unsignedBigInteger türünde bir sütun, ürünlerin ilişkilendirilmesi için kullanılır.
- `adet`: ürün adeti için integer türünde bir sütun.
- `timestamps()`: created_at ve updated_at sütunları otomatik olarak oluşturulur ve güncellenir.

2. `down()` fonksiyonu, `up()` fonksiyonunun geri alınması için kullanılır. `sepets` tablosu veritabanından silinir.

Bu migrasyon dosyası, `php artisan migrate` komutu çalıştırıldığında veritabanına `sepets` tablosunu ekler.

4) *Siparişler Tablosu*

Bu kod, Laravel Migration özelliğini kullanarak `siparislers` adında bir tablo oluşturur. Bu tabloda aşağıdaki sütunlar yer alır:

- `id`: siparişler tablosunun anahtar sütunu olarak tanımlanır ve her sipariş için benzersiz bir değer taşır.
- `kullanici_id`: siparişi veren kullanıcının kimliğini tutan yabancı anahtar sütunu, `kullanicis` tablosuna referans verir ve bir kullanıcı silindiğinde bağımlı siparişleri otomatik olarak siler (`onDelete('cascade')` özelliği sayesinde).
- `adres`: siparişin gönderileceği adresi tutar.
- `toplam_tutar`: siparişin toplam tutarını ondalık olarak tutar. Burada 10 haneli tam sayı ve 2 haneli ondalık kısım kullanılmaktadır.
- `durum`: siparişin durumunu belirten bir sütun olup varsayılan olarak 'Onay Bekliyor' olarak atanır.
- `timestamps`: `created_at` ve `updated_at` sütunlarını otomatik olarak oluşturarak her satırın oluşturma ve güncelleme tarihlerini tutar.

X. YAZILAN CONTROLLER SINIFLARI

A. *Controller Dosyası Oluşturmak*

Laravel'de, terminalden Controller oluşturmak için `make:controller` komutunu kullanabilirsiniz. Aşağıdaki adımları takip edebilirsiniz:

1. Terminale gidin ve projenizin kök dizinine ulaşın.

2. Aşağıdaki komutu girin:

```
php artisan make:controller ExampleController
```

Bu komut, `app/Http/Controllers` dizini altında `ExampleController.php` adında yeni bir Controller sınıfı oluşturur.

3. İsterseniz, Controller sınıfı oluşturulmadan önce `--resource` veya `--model` seçeneklerini kullanarak bir kaynak veya model Controller'ı oluşturabilirsiniz. Örneğin:

```
php artisan make:controller --resource ExampleController
```

Bu komut, `app/Http/Controllers` dizini altında `ExampleController.php` adında yeni bir kaynak Controller sınıfı oluşturur.

4. Yeni Controller'ınız şimdi oluşturuldu, artık bu sınıfı düzenleyebilirsiniz. Controller'ın yönlendirme işlemleri, form işlemleri ve veritabanı etkileşimleri gibi işlemleri yönetmesi beklenir.

B. *Controllerlerim*

1) *UserController*

Elbette, şimdi bahsettiğiniz fonksiyonları detaylı bir şekilde anlatalım.

`use App\Models\kullanici;` : Bu kullanım, Laravel'de kullanılacak olan "kullanici" model sınıfının tanımlanmasıdır. Model sınıfları, veri tabanındaki bir tablo ile ilişkili bir sınıfı temsil ederler ve bu sınıf aracılığıyla veri tabanına erişilir.

`use Illuminate\Support\Facades\Hash;` : Bu kullanım, Laravel'de şifreleme işlemleri için kullanılan Hash sınıfının tanımlanmasıdır. Hash sınıfı, güvenli bir şekilde şifrelenmiş verileri oluşturmak ve kontrol etmek için kullanılır.

`use Illuminate\Http\Request;` : Bu kullanım, Laravel'de HTTP istekleri ve yanıtları işlemek için kullanılan Request sınıfının tanımlanmasıdır.

`use Illuminate\Validation\Rule;` : Bu kullanım, Laravel'de doğrulama işlemleri için kullanılan Rule sınıfının tanımlanmasıdır. Rule sınıfı, doğrulama işlemlerinde kullanılan kural setlerini temsil eder.

`use Validator, Input, Redirect;` : Bu kullanım, Laravel'de doğrulama işlemleri için kullanılan Validator sınıfının, kullanıcı girdilerini almak için kullanılan Input sınıfının ve yönlendirmeleri yapmak için kullanılan Redirect sınıfının tanımlanmasıdır.

`use Illuminate\Support\Facades\DB;` : Bu kullanım, Laravel'de veri tabanı işlemleri için kullanılan DB sınıfının tanımlanmasıdır.

`public function verialma(Request $req) { ... }` : Bu fonksiyon, kullanıcının ad, soyad, telefon, adres, e-posta ve şifre bilgilerini alarak veri tabanına kaydetme işlemini gerçekleştirir. Veriler, kullanici model sınıfı aracılığıyla veri tabanına eklenir.

`function kontrol(Request $request) { ... }` : Bu fonksiyon, kullanıcının e-posta ve şifre bilgilerini doğrular. Önce doğrulama kurallarını belirlemek için Validator sınıfı kullanılır, ardından kullanıcı bilgileri kullanici model sınıfı aracılığıyla veri tabanından alınır. Eğer veri tabanında kullanıcının e-posta adresi bulunamazsa kullanıcı bilgileri hatalıdır ve hata mesajı döndürülür. Aksi takdirde, kullanıcının girdiği şifre, veri tabanındaki şifre ile karşılaştırılır ve doğruysa oturum başlatılır ve kullanıcının anasayfaya yönlendirilir. Hatalıysa, hata mesajı döndürülür.

`function kullanicibilgileriniyolla() { ... }` : Bu fonksiyon, oturum açmış kullanıcının bilgilerini alarak ana sayfaya yönlendiren fonksiyondur. `session()` yardımıyla, oturum açmış olan kullanıcının `id` bilgisi alınır ve bu bilgi kullanılarak, `kullanicis` tablosundan ilgili kullanıcının diğer bilgileri çekilir. `DB` sınıfı kullanılarak yapılan sorgu sonucunda, `kullanicis` tablosundan `id`'si oturum açmış kullanıcının `id`'sine eşit olan satır seçilir ve bu satırdaki tüm sütunlar alınarak `$k` değişkenine atanır.

Ayrıca, `veri` değişkeni içerisine `urunlers` tablosundaki tüm veriler alınır ve ana sayfaya yönlendirilirken `$veri` değişkeni ile birlikte gönderilir. Bu şekilde, ana sayfada oturum açmış kullanıcının bilgileri ve veritabanından alınan diğer veriler kullanılarak dinamik bir şekilde sayfa oluşturulur.

2) *ÜrünlerController*

Bu kodlar PHP dilinde yazılmış bir web uygulaması için kontrolör sınıfıdır. Bu kontrolör, bir ürün kataloğu ve sepet özelliklerini yönetmek için kullanılır.

Kodlarda, App\Http\Controllers ad alanı altında urunler adlı bir kontrolör sınıfı tanımlanmıştır. Bu sınıf, Controller sınıfından türetilmiştir.

Kontrolör, birkaç fonksiyon içermektedir:

verialma: Ürün eklemek için kullanılır. HTTP POST isteği ile gelen verileri alır, geçerlilik kontrolleri yapar ve veritabanına kaydeder.

guncelle: Mevcut bir ürünü güncellemek için kullanılır. HTTP POST isteği ile gelen verileri alır, veritabanında güncelleme işlemi yapar ve sonuç olarak ürünler sayfasına yönlendirir.

urungoster: Tüm ürünleri göstermek için kullanılır. Veritabanından tüm ürünleri alır ve satici.blade.php görünümüne aktarır.

urunler: Oturum açmış kullanıcının sahip olduğu ürünleri göstermek için kullanılır. Veritabanından sadece oturum açmış kullanıcının ürünlerini alır ve urunlerim.blade.php görünümüne aktarır.

incele: Bir ürünü ayrıntılı olarak incelemek için kullanılır. HTTP GET isteği ile gelen ürün kimliğini alır, veritabanından ürün bilgilerini alır ve incele.blade.php görünümüne aktarır.

sepeteEkle: Bir ürünü sepete eklemek için kullanılır. HTTP GET isteği ile gelen ürün kimliğini alır, sepet tablosuna kaydeder ve

sonuç olarak bir önceki sayfaya geri döner.

favori: Bir ürünü favorilere eklemek için kullanılır. HTTP GET isteği ile gelen ürün kimliğini alır, veritabanından ürün bilgilerini alır ve favori.blade.php görünümüne aktarır.

sil: Bir ürünü silmek için kullanılır. HTTP GET isteği ile gelen ürün kimliğini alır, veritabanından ürünü siler ve sonuç olarak bir önceki sayfaya geri döner.

3) SiparisController

Bu sınıf, Laravel framework'ünün `Controller` sınıfından türetilmiştir. Ayrıca `siparisler` modelini kullanmaktadır.

`post` fonksiyonu, HTTP POST isteği ile çağrıldığında çalışır ve `$request` parametresi ile isteğe ait verileri alır. Fonksiyon, bir sipariş kaydı oluşturarak, gerekli alanları atar ve kaydeder. Ayrıca, `session('LoggedUser')` ile oturum açmış kullanıcının kimliğini alarak `kullanici_id` alanına atar. Ardından, `DB::table('sepets')->where('sepet_id','=',session('LoggedUser'))->delete();` ile oturum açmış kullanıcının sepetini siler. Son olarak, `back()` fonksiyonuyla kullanıcının önceki sayfasına yönlendirir ve sipariş tamamlandı mesajını gösterir.

a) SepetController

`sepetim` fonksiyonu, kullanıcının sepetindeki ürünleri gösterir ve sepet içerisindeki her ürünün toplam adedini hesaplar. Ayrıca, kullanıcının adresini de veritabanından çeker ve bu verileri bir görünüme aktararak kullanıcıya sunar.

`sil` fonksiyonu ise sepetten bir ürünü siler. Kullanıcının sepetinden silmek istediği ürünün ID'si bu fonksiyona parametre olarak gönderilir. Sonrasında, veritabanından bu ürünün ID'sine göre kaydı silinir ve kullanıcı sepetine geri yönlendirilir.

XI. HTML SAYFALARI

A. Blade

Laravel'in Blade şablon motoru, Laravel'in MVC mimarisindeki View katmanında kullanılır. Blade, PHP dosyaları içinde yerleştirilen düzenli ifadelerin kullanılmasına izin veren bir şablon motorudur. Blade şablonlarında, HTML kodu içinde PHP kodu kullanılabilir ve bu PHP kodu, şablonların çıktısının derlenmesi sırasında işlenir.

Blade şablonlarında, HTML kodu ve PHP kodu ayrılmıştır ve HTML kodu daha okunaklı hale getirilir. Blade ayrıca, bir dizi kullanışlı özellik sağlar, örneğin şablon kalıtımı, şablon bileşenleri, şablon dahil etme, koşullu ve tekrar eden yapılar, form işleme ve daha fazlasını sağlar.

Blade şablonları, `.blade.php` uzantılı dosyalar olarak kaydedilir ve Laravel'in önceden tanımlanmış şablonlarını veya kendi özelleştirilmiş şablonlarınızı kullanabilirsiniz. Blade şablonlarında, HTML kodu ile birlikte `{{ }}` arasına alınmış PHP kodu kullanılır. Örneğin, `{{ $name }}` kodu, `$name` değişkeninin değerini HTML çıktısı içinde gösterir.

Blade ayrıca, if-else yapıları, for ve foreach döngüleri, include komutu, extends ve section kullanımı gibi birçok faydalı özelliği de destekler. Bu özellikler sayesinde, Blade şablonları, Laravel uygulamalarında hızlı ve esnek bir şekilde kullanılabilir.

B. Bootstrap

Bootstrap, HTML, CSS ve JavaScript kütüphanelerinden oluşan bir arayüz çatısıdır. Bootstrap kullanarak, web sitenize daha hızlı ve kolay bir şekilde, responsive ve modern bir tasarım sağlayabilirsiniz.

Bootstrap'ı HTML sayfanıza eklemek için aşağıdaki adımları izleyebilirsiniz:

1. Bootstrap kütüphanesini indirin: Bootstrap resmi web sitesinden indirilebilir. Alternatif olarak, CDN'den de yararlanabilirsiniz.

2. Bootstrap dosyalarını HTML sayfanıza ekleyin: Bootstrap kütüphanesini HTML sayfanıza eklemek için `link` ve `script` etiketlerini kullanabilirsiniz. CSS dosyalarını `head` etiketleri arasına, JavaScript dosyalarını ise `body` etiketlerinin en altına eklemelisiniz.

Bootstrap kullanarak HTML sayfanızı oluşturun: Bootstrap, HTML elementleri için önceden tasarlanmış sınıflar sağlar. Örneğin, `container` sınıfı, sayfanızın içeriğinin tam ortasına hizalanmasını sağlar. `row` sınıfı, içeriği sütunlara bölmek için kullanılır. `col` sınıfı ise sütunların boyutunu belirler.

C. Layout

Bir sayfanın düzenini oluşturmak için genellikle belirli bir yapı kullanılır. Örneğin, bir header, bir footer ve sayfa içeriği gibi bileşenler genellikle aynı düzeni paylaşırlar. Bu bileşenleri oluşturmak ve diğer sayfalarda yeniden kullanmak için include yöntemi kullanılabilir.

Daha önce oluşturduğumuz header, footer ve nav kısımlarını include etmek için PHP'nin include() veya require() fonksiyonlarını kullanabiliriz. Bu fonksiyonlar, belirtilen dosyayı mevcut dosyaya dahil eder.

Örneğin, header dosyasını dahil etmek için:
<?php include('header.php'); ?>

veya
<?php require('header.php'); ?>

Footer ve nav dosyalarını da aynı şekilde dahil edebiliriz. Bu sayede, sayfa düzenimizi tekrar tekrar yazmak yerine, bu dosyaları diğer sayfalarda da kullanabiliriz.

Bu sayede, header, nav ve footer dosyalarındaki değişiklikler otomatik olarak tüm sayfalara yansıyacaktır.

D. Web Sayfaları

1) İlk Sayfa

Bu sayfada ilk olarak, `session_start()` çağrısı ile bir oturum başlatılır ve kullanıcı istediği takdirde `$_GET` parametresi kullanılarak `logout` değişkeni "true" olarak ayarlanabilir ve `session_destroy()` çağrısı ile oturum sonlandırılabilir.

HTML belgesi, birkaç dış kaynakla birlikte, üstbilgi, navigasyon ve ana içeriği içeren birkaç bölüme ayrılmıştır. Üstbilgi, tarayıcının dil ayarlarını ve sayfanın genel görünüm ayarlarını içerir. Navigasyon, kullanıcının sayfalar arasında gezinmesine olanak tanıyan bir menüdür. Ana içerik, sayfanın geri kalanında görüntülenecek olan içeriktir.

`@csrf` ifadesi, Cross-Site Request Forgery'ye karşı bir koruma sağlamak için Laravel'de kullanılan bir araçtır.

Sayfanın stilini kontrol eden birkaç stil etiketi ve bir JavaScript dosyası da vardır.

Ayrıca, sayfa üzerinde bir giriş formu da vardır. Kullanıcı adı ve şifre bilgilerini alır ve `route()` fonksiyonu ile işleme yönlendirir. Formda ayrıca, bir önceki giriş denemesinin başarısız olduğunu belirten bir hata mesajı görüntülenebilir.

2) Kullanıcı Anasayfa

Bu sayfada Özet olarak, bir foreach döngüsü ile veritabanından çekilen ürün verilerini kullanarak bir ürün kartı oluşturur. Her bir ürün için bir div elementi açılır ve içine bootstrap sınıflarıyla kart yapısı eklenir. Kartın üst kısmına, ürünün görselini gösteren bir img elementi yerleştirilir. Kartın gövdesinde, ürün adı, markası ve fiyatı gibi bilgiler bir h5, p ve span elementleri ile gösterilir. Son olarak, kartın alt kısmında, ürünü incelemek veya sepete eklemek için iki buton eklenir. Bu butonlar, kullanıcının ilgili sayfaya yönlendirilmesini sağlamak için href attribute'ları ile ilgili route'lara yönlendirilir.

3) Ürün İnceleme Sayfası

Bu sayfa, ürünün resmini, adını, markasını, modelini, kategorisini, fiyatını, satıcısını, ürün hakkında bilgiyi ve bir butonu içerir. Buton, sepete eklemek için kullanılır. Ayrıca, butona tıklandığında sepete eklenen ürünün id'si, bir PHP oturumu kullanılarak saklanır.

4) Sepetim Sayfası

Sayfa, kullanıcının sepetindeki ürünleri listeleyen bir tablo ve sipariş özetini gösteren bir karttan oluşuyor. Ayrıca, kullanıcının siparişini tamamlamak için kullanabileceği bir form da var.

Sayfa, Laravel web framework'ünün bir parçası olan Blade template engine'i kullanılarak oluşturuluyor. İlk olarak, veritabanından kullanıcının sepetindeki ürünleri çekmek için kullanılan bir sorgu var. Bu sorgu, "urunler" adlı bir değişken içinde saklanan verileri döndürüyor. Daha sonra, bu veriler, foreach döngüsü kullanılarak bir tablo içinde listeleniyor. Tablo hücreleri, verileri ekrana yazdırmak için Blade template engine'inin özellikleriyle dolduruluyor.

Ayrıca, her ürün için ürün detaylarını almak için bir sorgu yapılıyor. Bu sorgu, "urun_detay" adlı bir değişkene atanıyor ve tablo içinde kullanılıyor. Ayrıca, ürünlerin toplam fiyatını hesaplamak için bir döngü kullanılıyor ve sonuç, "toplam_fiyat" adlı bir değişkene atanıyor. Sipariş tamamlandığında kullanılacak form, ürünleri sipariş eden kullanıcının adresini ve toplam fiyatı gönderiyor.

Bu kod, web sitesinde ürünlerin listelenmesi ve müşterilerin sipariş tamamlama işlemini gerçekleştirmelerine olanak tanıyan temel bir altyapı sağlıyor.

Bu sayfada, bir kullanıcının sepetindeki ürünleri ve sipariş özetini gösteren bir HTML tablosu oluşturuluyor. Bu işlem sırasında veritabanından ürünlerin fiyatları ve adetleri alınıyor ve tabloya ekleniyor. Ayrıca, sipariş tamamlama işlemi için bir form da oluşturuluyor.

Kodlar, Laravel PHP framework'ü ile yazılmıştır ve bu framework'ün sunduğu bazı fonksiyonları kullanıyor. `DB` facade'i, veritabanı işlemlerini kolaylaştırmak için kullanılır.

Kodların ayrıntılı açıklaması şöyledir:
- `use Illuminate\Support\Facades\DB;`: Veritabanı işlemleri yapmak için Laravel'in sağladığı DB sınıfını kullanmak için `use` ifadesi ile eklenir.
- `if(count($urunler) > 0)`: Eğer sepet boş değilse, ürünleri tabloya eklemek için bir döngü oluşturulur.
- `$urun_detay = DB::table('urunlers')->where('id', '=', $urun->urun_id)->first();`: `$urun_detay` değişkenine, `$urun` değişkeninde tutulan ürünün detaylarını veritabanından almak için bir sorgu yapılır.
- `<td>{{ $urun_detay->urunadi }}</td>`: `$urun_detay` değişkenindeki ürünün adını ekrana yazdırmak için Blade templating motoru kullanılır.
- `<?php $toplam_fiyat = 0;`: `$toplam_fiyat` değişkeni, ürünlerin toplam fiyatını hesaplamak için tanımlanır.
- `$urun_detay = DB::table('urunlers')->where('id', '=', $urun->urun_id)->first();`: Her bir ürün için fiyatı veritabanından alınır.
- `<td name="toplam_fiyat">₺{{ $toplam_fiyat }}</td>`: Toplam fiyatı ekrana yazdırmak için Blade templating motoru kullanılır.
- `<input type="hidden" name="adres" value="{{ $kullanici }}">`: Kullanıcının adres bilgisini, siparişi tamamlama formuna saklamak için gizli bir input oluşturulur.
- `<button type="submit" class="btn btn-primary float-end">Siparişi Tamamla</button>`: Siparişi tamamlama işlemi için bir submit button'u oluşturulur.

5) Kullanıcının Siparişleri Sayfası

Bu sayfada bir kullanıcının veritabanındaki siparişlerini görüntülemek için bir sorgu oluşturuldu. Bu sorgu, "siparislers" tablosunda kullanıcı kimliğine göre siparişleri seçer ve ardından "foreach" döngüsü kullanarak her siparişi bir "card" bileşeninde görüntüler. Sipariş numarası, toplam tutar, tarih ve durum gibi bilgiler, her bir "card" bileşeninde ayrı bir etiketle gösterilir. Durum etiketi, siparişin hangi aşamada olduğunu belirtir ve farklı renklerde (örneğin, mavi, turuncu, yeşil, kırmızı) gösterilir. Bu sayfa, bir e-ticaret sitesinde müşterilerin sipariş geçmişlerini görüntülemek için kullanılabilir.

6) Kullanıcı Profili Sayfası

Bu sayfada, oturum açmış bir kullanıcının bilgilerini alır. `session_start()` fonksiyonu, PHP oturumunu başlatır ve `DB::table('kullanicis')` ile veritabanından `kullanicis` tablosunu seçer. Ardından `where('id', session('LoggedUser'))` ile oturum açmış kullanıcının ID'sine göre bir sorgu yapar. Sorgu sonucu `$kullanicilar` değişkenine atanır ve `@foreach` döngüsüyle her bir kullanıcının bilgileri ekrana yazdırılır. `<?php echo $kullanici->kullaniciAdi ?>` vb. ifadelerle, kullanıcının adı, soyadı, e-posta adresi, adresi ve telefon numarası gibi bilgileri ekranda gösterilir. Son olarak, tüm bu kod bloğu bir `center` etiketi içinde yer alır ve bootstrap kütüphanesiyle tasarlanmış bir kart kullanarak bilgileri düzenli bir şekilde gösterir.

7) *Satıcının Anasayfası*

Bu sayfa HTML sayfasındaki bir ürün listesini göstermek için kullanılır. Ürün listesi, veritabanından alınan ürün bilgilerini (ürün adı, marka, model, kategori, fiyat, satıcı ve açıklama) ve her bir ürünün resmini içerir.

İlk satır, CSS kodunu içerir ve burada "small-card" sınıfı, ürün kartları için bir maksimum genişlik ve ortalanma özelliği ayarlar.

Daha sonra, bir "row" öğesi oluşturulur ve burada tüm ürün kartları yerleştirilir. "Veri" adlı bir döngü, her bir ürün için bir kart oluşturmak için kullanılır. Bu döngü "col-lg-3 col-md-6 mb-4" sınıfı içinde yer alır, bu da her bir ürünün 4 sütun genişliğine sahip olmasını ve bir mobil cihazda 6 sütun genişliğinde görünmesini sağlar.

Her bir ürünün kartı bir "card" öğesi içinde oluşturulur. Resim, "card-img-top" sınıfı kullanılarak yerleştirilir. Ürün adı, "card-title" sınıfı kullanılarak eklenir. Fiyat, "badge" ve "rounded-pill" sınıflarını kullanarak bir kenarlık eklenerek ve "bg-warning" ve "text-dark" sınıflarını kullanarak sarı bir arka plan ve koyu bir metin rengiyle birlikte görüntülenir.

Tüm ürün kartları "product-dom" öğesi içinde yer alır ve "p-5" sınıfı kullanılarak bir iç boşluk verilir. Son olarak, tüm tablo kodu kapatılır.

8) *Satıcı Ürün Ekle&Düzenle Sayfası*

Bu sayfada, bir e-ticaret sitesinde ürün listesi gösteriliyor. Sayfanın en üstünde "Ürün Ekle" butonu bulunuyor ve bu butona tıklandığında "edit_form" isimli bir div açılıyor. Bu div içinde, bir ürünün özelliklerini (ürün adı, marka, model, fiyat, satıcı ve kategori) girerek, yeni bir ürün eklemek mümkün oluyor.

Aşağıda ise, daha önceden eklenmiş olan ürünlerin listesi yer alıyor. Bu listede her bir ürünün resmi, adı, markası, modeli, fiyatı, kategorisi ve satıcısı gösteriliyor. Her bir ürünün yanında "Güncelle" ve "Sil" butonları bulunuyor. "Güncelle" butonuna tıklandığında, o ürünün bilgilerini düzenleyebileceğimiz bir form açılıyor. "Sil" butonuna tıklandığında ise, o ürünü silebileceğimiz bir onay sayfasına yönlendiriliyoruz.

9) *Satıcının Sipariş Durumu Sayfası*

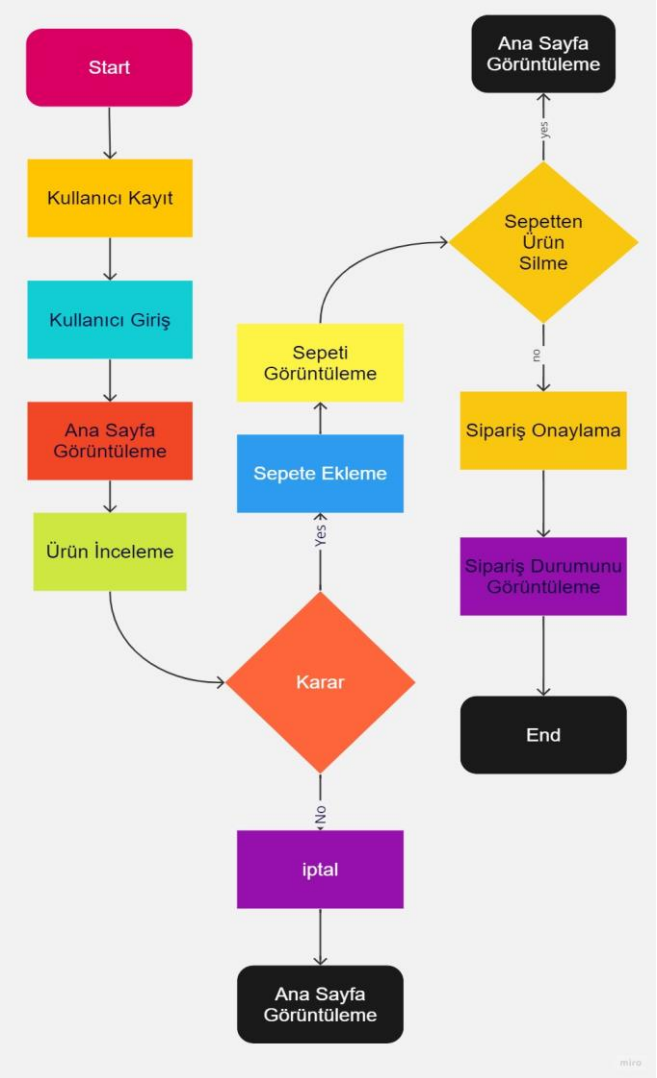
Bu sayfa, bir veritabanı tablosundaki siparişlerin bir listesini görüntüler ve siparişlerin durumunu değiştirmek için düğmeler sağlar.

Sayfada bir tablo görüntülenir. Her satır, siparişlerin farklı özelliklerini (sipariş ID'si, kullanıcı ID'si, adres, toplam tutar ve durum) gösterir. Her sipariş için, "Onaylandı", "İptal Edildi", "Yola Çıktı" ve "Teslim Edildi" durumlarını değiştirmek için düğmeler vardır.

Kodda, önce bir veritabanı bağlantısı oluşturulur ve "siparislers" tablosundaki tüm siparişler sorgulanır. Daha sonra, her sipariş için bir satır oluşturulur ve sipariş özellikleri tabloda görüntülenir. Son olarak, her sipariş için durum değiştirme düğmeleri oluşturulur ve düğmelerin tıklanması durumunda durum değişikliği veritabanına kaydedilir ve sayfa yeniden yönlendirilir.

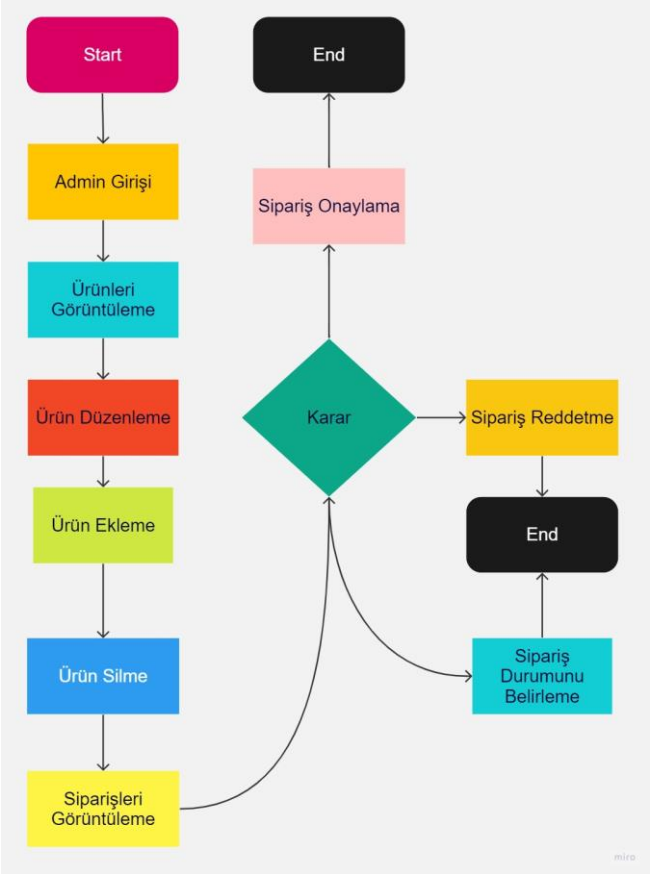
Bu kodu bir rapora eklemek istiyorsanız, sipariş yönetim sistemi hakkında bir rapor yazıyorsanız veya bir web uygulaması geliştiriyorsanız, bu kodu kullanabilirsiniz. Ancak, kodu kullanmadan önce, veritabanı bağlantısı için gerekli bilgileri sağlamalısınız ve tablo adını, sütun adlarını ve veri türlerini değiştirmeniz gerekebilir. Ayrıca, sayfa tasarımınızla uyumlu hale getirmek için CSS kodlarını eklemeniz gerekebilir.

XII. AKIŞ DIYAGRAMI



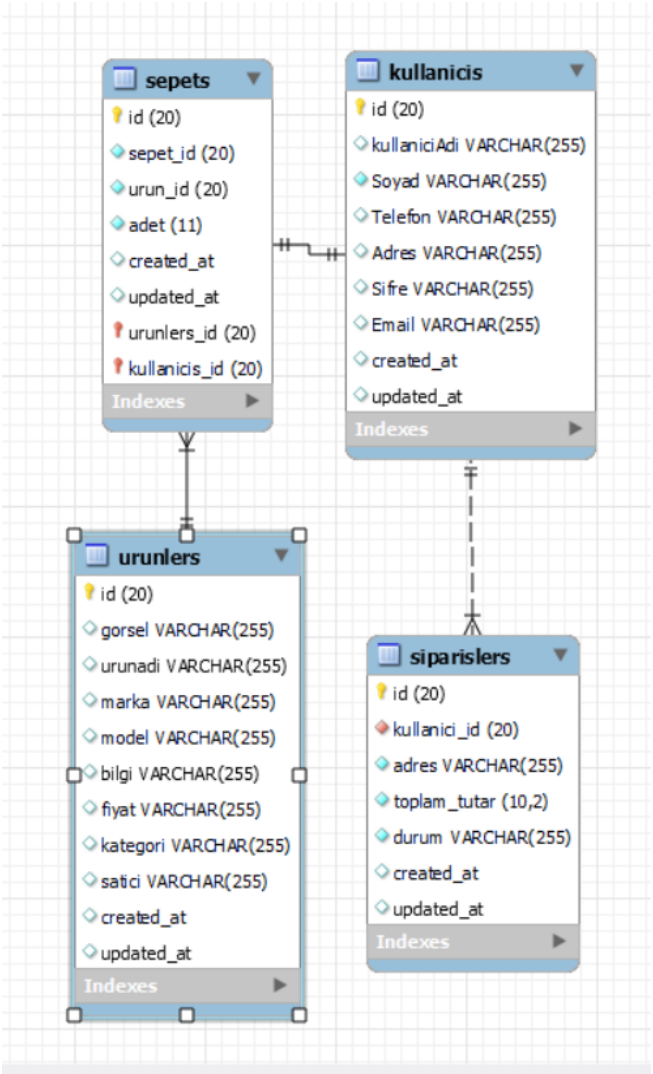
İşlem	Açıklama
Kullanıcı Kayıt	Kullanıcı, siteye kayıt olur. Kayıt formunda, kullanıcı adı, e-posta adresi ve şifre istenir. Kullanıcı bilgileri, veritabanına kaydedilir.
Kullanıcı Giriş	Kullanıcı, kayıt olduktan sonra, kullanıcı adı ve şifresi ile siteye giriş yapar. Kullanıcı bilgileri, veritabanından kontrol edilir. Doğru bilgiler girilirse, kullanıcı ana sayfaya yönlendirilir.
Ana Sayfa Görüntüleme	Kullanıcı, ana sayfada yer alan ürünlerin listesini görüntüler. Ürünlerin yanında "İncele" ve "Sepete Ekle" butonları bulunur.
Ürün İnceleme	Kullanıcı, ana sayfada yer alan bir ürünün detaylı bilgilerini görüntüler. Ürün sayfasında "Sepete Ekle" butonu yer alır.
Sepete Ekleme	Kullanıcı, ürün sayfasındaki "Sepete Ekle" butonuna tıklayarak, sepetine ürün ekler.
Sepeti Görüntüleme	Kullanıcı, sepetindeki ürünleri görüntüler. Sepet sayfasında, her ürün için "Sil" ve "Onayla" butonları yer alır.
Sepetten Ürün Silme	Kullanıcı, sepet sayfasında yer alan "Sil" butonuna tıklayarak, sepetinden bir ürünü siler.

Sipariş Onaylama	Kullanıcı, sepet sayfasında yer alan "Onayla" butonuna tıklayarak, sepetindeki ürünleri satın alır. Sipariş, veritabanına kaydedilir.
Sipariş Durumunu Görüntüleme	Kullanıcı, "Siparişlerim" sayfasında, siparişlerinin durumunu görüntüler.



İşlem	Açıklama
Admin Girişi	Admin, siteye kullanıcı adı ve şifresi ile giriş yapar.
Ürünleri Görüntüleme	Admin, anasayfada bulunan ürünleri görüntüler.
Ürün Ekleme	Admin, ürün ekleme sayfasından yeni bir ürün ekler.
Ürün Düzenleme	Admin, mevcut ürünlerin listelendiği sayfadan istediği ürünü seçerek düzenler.
Ürün Silme	Admin, mevcut ürünlerin listelendiği sayfadan istediği ürünü seçerek siler.
Siparişleri Görüntüleme	Admin, siparişler sayfasından siparişleri görüntüler.
Sipariş Onaylama	Admin, siparişler sayfasından siparişleri onaylar.
Sipariş Reddetme	Admin, siparişler sayfasından siparişleri reddeder.
Sipariş Durumunu Belirleme	Admin, siparişler sayfasından sipariş durumunu belirler.

XIII. VARLIK İLİŞKI DIYAGRAMI



XIV. SONUÇ

Laravel, PHP programlama dilinde geliştirilmiş bir web uygulama çatısıdır. Bu raporda, Laravel ile ilgili temel kavramlar ve işlemler h akkında bilgi verildi. MySQL veri tabanı bağlama işlemleri, Laravel içinden tablo oluşturma işlemleri, Laravel'in Controller sınıfını kullanma işlemleri, Laravel'in Migration sınıfını kullanma işlemleri ve Laravel'in Model dosyalarını kullanma işlemleri anlatıldı. Ayrıca, HTML sayfalarının düzeni ve Controller sınıfında yazılan fonksiyonlar ve web.php dosyasına route ayarlama işlemlerinden de bahsedildi.

Laravel, web uygulamaları geliştirmek için kullanılan popüler bir çatıdır ve web uygulamalarının geliştirilmesini kolaylaştırır. Bu raporda, Laravel ile ilgili temel kavramlar ve işlemler hakkında bilgi verilerek, web uygulamaları geliştirmek isteyenlerin işlerini kolaylaştıracak bir kaynak sunulmuştur.

XV. KAYNAKÇA

[3] "Laravel Dokümantasyonu." Laravel.com, Taylor Otwell, 2023. Erişim tarihi: 7 Mayıs 2023. https://laravel.com/docs/8.x.