

**BURSA TEKNİK ÜNİVERSİTESİ**  
**MÜHENDİSLİK VE DOĞA BİLİMLERİ FAKÜLTESİ**  
**BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**  
**SEMİNER DERSİ RAPORU**

**LUMEN İLE REST API GELİŞTİRME**

Hüdanur EDİZ

2022-2023 Bahar Dönemi

# ÖNSÖZ

Bursa Teknik Üniversitesi Bilgisayar Mühendisliği bölümü 3. Sınıf Bahar Dönemi Seminer dersi kapsamında oluşturulan bu rapor, açık kaynaklı Lumen frameworku kullanılarak REST API oluşturma aşamalarını tanımlayan bir kılavuz olarak hazırlanmıştır.

HÜDANUR EDİZ

Bursa 2023

# **İÇİNDEKİLER**

<b>ÖNSÖZ.....</b>	<b>2</b>
<b>İÇİNDEKİLER.....</b>	<b>3</b>
<b>1.Genel Bilgiler.....</b>	<b>4</b>
1.1 Lumen Nedir?.....	4
1.2. Rest Mimarisi Nedir?.....	6
1.3. HTTP İstek ve Durum Kodları Nelerdir?.....	7
<b>2. Lumen Projesi Oluşturma ve Yapılandırma.....</b>	<b>9</b>
2.1. Lumen Gereksinimleri Nelerdir?.....	9
2.2 Lumen Kurulumu.....	14
2.3 Lumen Konfigürasyonları.....	17
<b>3. CRUD İşlemlerini Gerçekleştiren Rest API Oluşturma.....</b>	<b>20</b>
3.1. CRUD İşlemleri Nelerdir?.....	20
3.2. Route Tanımlamaları.....	20
3.3. Migration Oluşturma.....	21
3.4. Controller Dosyası Oluşturma.....	23
3.5. Seeder Kullanımı.....	26
<b>4. API Test İşlemleri.....</b>	<b>28</b>
4.1 Postman Nedir?.....	28
4.2 Postman ile API Testleri.....	28

## 1. Genel Bilgiler

### 1.1. Lumen Nedir?

Lumen açık kaynak kodlu micro php frameworkudur. API işlemleri ve mikro işlemler için tasarlanan lumen, Laravelin geliştiricisi olan Taylor Otwell tarafından geliştirilmiştir ve 2015 yılında MIT lisansı altında tanıtılmıştır. Laravel'in sadeleştirilmiş versiyonu olan lumen, API işlemlerini hızlı gerçekleştirmek için kullanılır. Bu özellik sayesinde response dönüşleri hızlıdır. Ayrıca Laravel projesine yükseltilebilir ve bu işlem oldukça kolay bir şekilde yapılır. Bu özellik kapsamı büyüyebilecek projelerde oldukça avantaj sağlar.



# Lumen

Symfony frameworklerin tamamı doğrudan lumen frameworkunde mevcut değildir. Bunun sebebi API geliştirmek için kullanılan lumende bazı symfony bileşenlerine gerek duyulmayacağı düşüncesidir. Yer almayan bileşenlerden biri form bileşenidir. Form işlemlerini sağlayan bu bileşen yer almadığından Symfony form özellikleri kullanılamaz ancak form işlemleri manuel gerçekleştirilebilir.

Security bileşeni de lumende mevcut olmayan bileşenlerden biridir. Lumen Kendi güvenlik sistemini kullanır.

Çoklu dil desteği içeren Symfony Translation bileşeni de bu framework kapsamında değildir.

Oturum işlemlerinde kullanılan Session yapısı da mevcut özelliklerden değildir. Bu sebeple farklı yollardan oturum işlemleri yönetilir.

Cache bileşeni Lumen'de dolaylı yoldan bulunur. Laravel'de bulunan cache componenti kullanılır.

Lumen frameworku küçük ve orta ölçekli projelerde tercih edilir. Benzerleri olan slim ve slex gibi micro-frameworklere göre daha performanslı ve hızlıdır.

**Routing**(yönlendirme), **database abstraction** (veri tabanı soyutlama), **queueing** (kuyruk işlemleri), **caching** (ön belleğe alma) gibi işlemleri kolaylıkla gerçekleştirmeyi sağlar. Ayrıca Http protokolleri için destek sağlar.

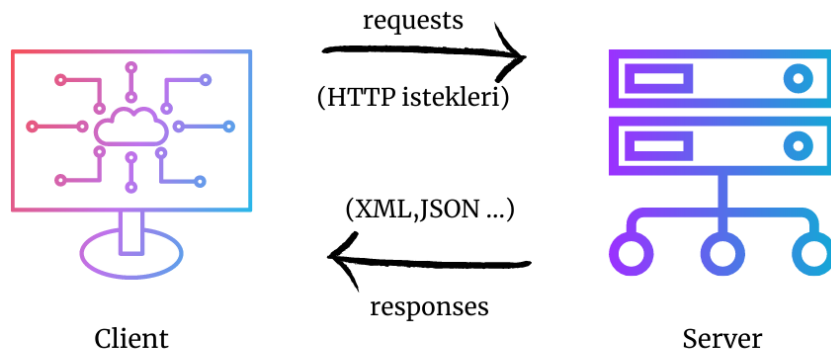
## 1.2. REST Mimarisi Nedir?

REST'in açılımı Representational State Transferdir ve Temsili Durum Transferi anlamına gelmektedir. 2000 yılında Roy Fielding tarafında doktora tezi kapsamında öne sürülmüştür. Web servisler için tasarlanmıştır. İstemci(client) -sunucu(server) arası iletişimi Http (Hypertext Transfer Protocol) protokolü ile sağlar. Bu iletişim url adresleri üzerinden taşınarak gerçekleştirilir. Client tarafında gönderilen request(istek) sonucu bir response (yanıt) döner. Requests sonucunda dönen response XML, JSON veya TEXT formatında veri döndürülür.

Platformlar arası uyumludur. Veri alışverişinde kolaylık sağlar. Durumsuzdur yani alınan her mesaj hem istemci hem sunucu tarafında bir ön koşul olmaksızın yorumlanabilir.

REST API, Rest mimarisi ile oluşturulan API (Uygulama Programlama Arayüzü)'lerdir. HTTP protokolü üzerinden veri alışverişi sağlanır. İstemci(client) tarafından sunucuya(server) HTTP isteklerinde (GET, POST, PUT, DELETE) bulunulur. Url'ler üzerinden gönderilen İsteklerin sunucuya ulaşması (ya da ulaşmaması) durumunda response döndürülür.

Aşağıda bulunan görselde bir REST API'nin temel çalışma mantığı görüntülenmektedir.



**Şekil 1- Rest API çalışma mantığı**

İstek sonucunda oluşan responselar durum mesajları ile döndürülür. Bu durum mesajları HTTP durum kodları olarak isimlendirilmektedir.

### 1.3. HTTP İstek Türleri ve Durum Kodları Nelerdir?

HTTP (Hypertext Transfer Protocol) istemci ve sunucu arasında iletişim kurmayı sağlayan protokoldür. İstemci, sunucuya HTTP isteklerinde bulunarak erişmek istediği kaynak (web sayfaları, API ...) ile iletişim kurmaya çalışır ve bu iletişimin sağlanması sonucunda bir response (yanıt,dönüt) alır. REST API oluştururken sunucu ile bu HTTP istekleri üzerinden veri akışı sağlanır.

Sık kullanılan 4 çeşit http istek türleri şunlardır:

- **GET:** Sunucudan veri almak, görüntülemek için kullanılır.
- **POST:** Yeni bir veri/kaynak oluşturmak için sunucuya veri gönderir.
- **PUT:** Kaynak /veri güncelleme sağlar. Güncellenecek bölüme ait tüm veriler parametre olarak verilmeli ve sunucuya aktarılmalıdır.
- **DELETE:** Sunucudan veri silme işlemini gerçekleştiren istektir.

İstek sonucunda dönen responselarda durum kodları yer almaktadır. Bu durum kodlarının bazıları ve açıklamaları aşağıdaki tabloda yer almaktadır.

1xx durum kodları: Bilgilendirme	
100(Continue)	İstemci tarafından gönderilen istek sunucuya başarılı bir şekilde ulaşmıştır. İstek işlemi devam etmektedir.
101(Switching Protocols)	Protokol değişim durumunda kullanılır. Sunucu tarafından desteklenmeyen protokoller istemciye bildirilerek başka bir protokole (Genellikle web socket bağlantısı) geçiş yapılır.
102(Processing)	İstek sunucu tarafından alınmıştır ve işlenmeye devam etmektedir.
2xx durum kodları: Başarılı	
200 (OK)	İsteğin başarılı olduğunu belirtir. Giden istek başarılı bir şekilde sunucuya ulaşmış ve cevap dönmüştür.
201(Created)	İstek sonucunda öge oluşumunun başarılı bir şekilde gerçekleştiğini belirtir. Genellikle POST işlemleri
203 (Non-Authoritative Information)	İstek başarılıdır fakat oluşan response sunucunun yetkisi olmadığına istemciye ulaşmadan önce bir dönüştürücü tarafından değiştirilmiştir.
204(No Content)	İsteğin başarılı bir şekilde gerçekleştiğini ancak response'da bir içerik bulunmadığını ifade eder.
3xx durum kodları: Yönlendirme	
301(Moved Permanently)	İsteğin kalıcı olarak başka bir adrese taşındığını ifade eder. Bu kodda bulunan Location yeni adrese yönlendirmeyi sağlar.

302(Found)	İsteğin geçici olarak başka bir adrese taşındığını ifade eder. Bu kodda bulunan Location geçici adrese yönlendirmeyi sağlar.
304(Not Modified)	İstemci tarafından daha önceden yapılan GET isteği sonrasında herhangi bir farklılık olmadığı durumlarda kullanılır. Önbellekte bulunan verilerin kullanılması gerektiğini belirtir.
4xx durum kodları: İstemci Hatası	
400 (Bad Request)	İstemci tarafından geçersiz istek gönderildiğini belirten durum kodudur.
401 (Unauthorized)	Erişim yetkisi olmadığı ve doğrulama sağlanmadığını ifade eder.
404 (Not Found)	İstenilen kaynak bulunamadığında dönen hata mesajıdır.
408(Request Timeout)	İstek zaman aşımına uğradığında dönen durum mesajıdır.
5xx durum kodları: Sunucu Hatası	
500 (Internal Server Error)	Sunucu kaynaklı hatalar sonucu dönen durum mesajıdır. Veri tabanı sorunları ya da sunucu erişim problemleri kaynaklı hatalar olabilir.
503 (Service Unavailable)	Sunucunun geçici olarak hizmet veremeyeceğini belirtir.
505(HTTP Version Not Supported)	İsteğin sunucu tarafından desteklenmeyen bir HTTP protokolü ile yapıldığını belirtir.

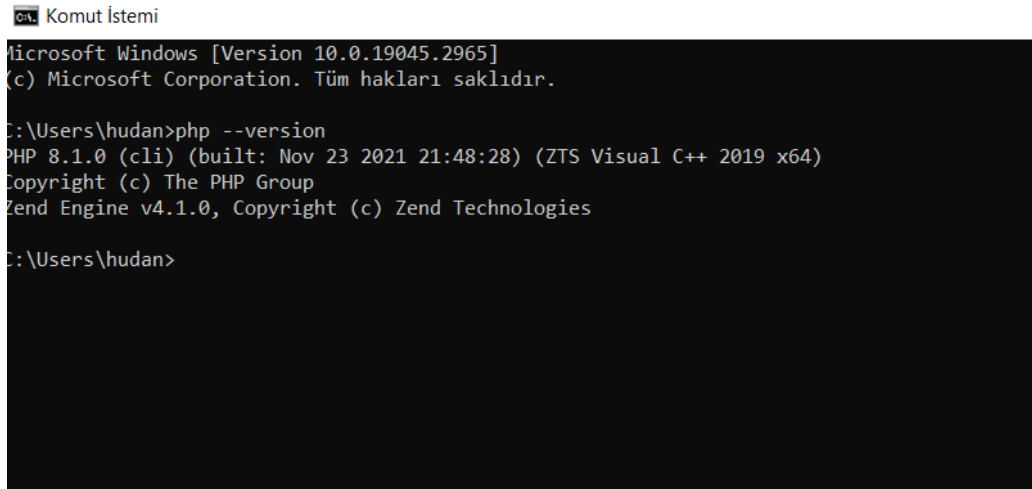


## 2. Lumen Projesi Oluřturma ve Yapılandırma

### 2.1. Lumen Gereksinimleri Nelerdir?

Lumen micro-frameworkunu kullanmak için bazı gereksinimlere ihtiyaç vardır. Bu gereksinimler kullanılan lumen versiyonuna göre deęişiklik gösterebilir. Son sürüm olan 10.x versiyonu için gereksinimler řu řekildedir:

- **PHP sürümü:** Sistem üzerinde kurulu olan PHP sürümü 8.0 ve üzerinde olmalıdır.



```
Komut İstemi
Microsoft Windows [Version 10.0.19045.2965]
(c) Microsoft Corporation. Tüm hakları saklıdır.

C:\Users\huda>php --version
PHP 8.1.0 (cli) (built: Nov 23 2021 21:48:28) (ZTS Visual C++ 2019 x64)
Copyright (c) The PHP Group
Zend Engine v4.1.0, Copyright (c) Zend Technologies

C:\Users\huda>
```

*Şekil 2- PHP versiyon kontrolü*

- **Web Sunucusu:** Lumen projesi çalıştırmak için bir web sunucu gereklidir. Yerel geliştirme ortamında(local) çalışılıyorsa PHP dahili web sunucusu tercih edilebilir. Bu sunucu PHP 5.4 sürümü ve sonrasındaki sürümlerde mevcut olarak gelir. Bu web sunucusunu kullanmak için terminal açılır ve projenin bulunduğu konuma gidilir.

**php -S localhost:8000 -t public** komutu ile proje başlatılır.

```
C:\Users\hudan\OneDrive\Masaüstü>cd seminer
C:\Users\hudan\OneDrive\Masaüstü\seminer>cd restapiwithlumen
C:\Users\hudan\OneDrive\Masaüstü\seminer\restapiwithlumen>php -S localhost:8000 -t public
[Sun May 21 01:04:43 2023] PHP 8.1.0 Development Server (http://localhost:8000) started
```

**Şekil 3- PHP Dahili web sunucusunu çalıştırma**

Proje başlatıldıktan sonra görüntülenen ekran aşağıdaki şekildedir.



**Şekil 4- Projenin tarayıcı üzerinde görüntülenmesi**

Sunucuya gelen istekler ve yanıtlar terminalde aşağıdaki şekilde görüntülenir. Bu durum sayesinde istekler(request) ve yanıtlar(response) kolaylıkla takip edilebilir ve test işlemleri hızlıca gerçekleştirilebilir.

```
[Sun May 21 01:04:43 2023] PHP 8.1.0 Development Server (http://localhost:8000) started
[Sun May 21 01:06:28 2023] [::1]:54544 Accepted
[Sun May 21 01:06:28 2023] [::1]:54545 Accepted
[Sun May 21 01:06:31 2023] [::1]:54544 [200]: GET /
[Sun May 21 01:06:31 2023] [::1]:54544 Closing
[Sun May 21 01:06:31 2023] [::1]:54545 [404]: GET /favicon.ico - No such file or directory
[Sun May 21 01:06:31 2023] [::1]:54545 Closing
```

**Şekil 5- Terminalde görüntülenen istek ve yanıt durumları**

Üretim ortamında ise daha gelişmiş web sunucuları olan **Nginx** veya **Apache** gibi ölçeklendirilebilir, yüksek güvenlik ve performanslı sunucular tercih edilebilir.

- **Composer**: PHP paket yönetimi composer ile gerçekleşir. Lumende de diğer php frameworklerinde olduğu gibi proje kurulumu composer üzerinden gerçekleştirilir. Bu sayede tüm bağımlılıklar kurulu bir şekilde proje oluşturulmuş olur.

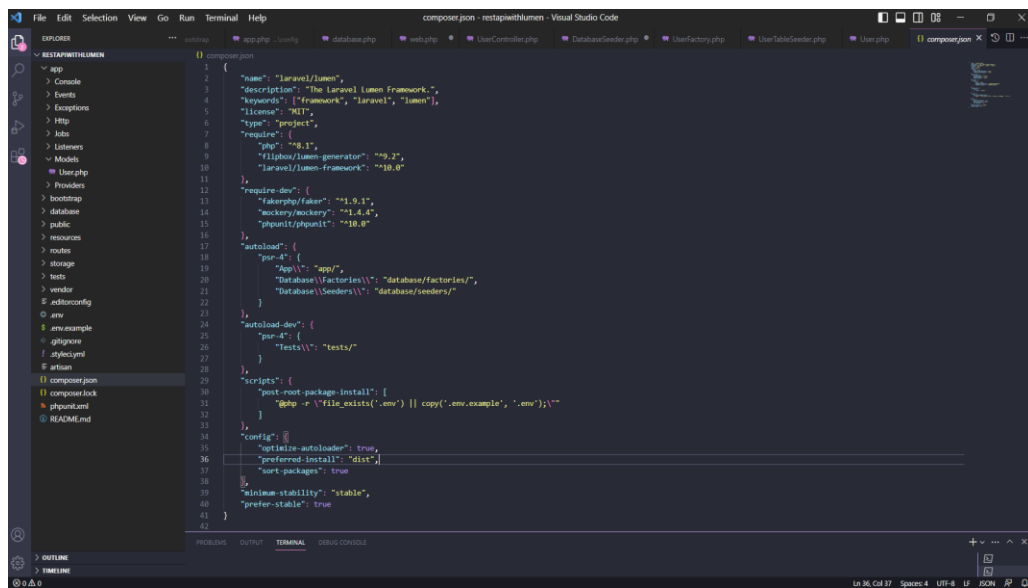
Projede bulunan '**composer.json**' dosyasında gerekli bağımlılıklar ve kütüphaneler tanımlanır. Composer ile gereken paketler bu dosyaya yüklenerek projeye yer alır.

Proje ilk oluşturulduğunda, **composer install** komutu ile gerekli bağımlılıklar projeye yüklenir ve composer kurulumu projede gerçekleşmiş olur. Yüklenen bağımlılıklar **composer.json** dosyasında bulunur. Gerektiğinde bağımlılıklarda değişiklik yapmak gerekebilir, bu dosya üzerinden değişiklikler yapılarak bağımlılıklar tekrar düzenlenebilmektedir.

Bağımlılıklarda güncelleme işlemleri **composer update** komutu ile yapılır. Bu komut sayesinde mevcut bağımlılık ve kütüphanelerin güncel versiyonları projeye dahil edilmiş olur.

Herhangi bir paket projeye eklenmek istendiğinde ise **composer require** komutu ile gereken bağımlılık projeye eklenir.

Aşağıdaki ekran görüntüsünde **composer.json** dosyasında bağımlılıkların nasıl tanımlandığı görüntülenmektedir.



Şekil 6- composer.json dosyasında bağımlılık tanımlanması

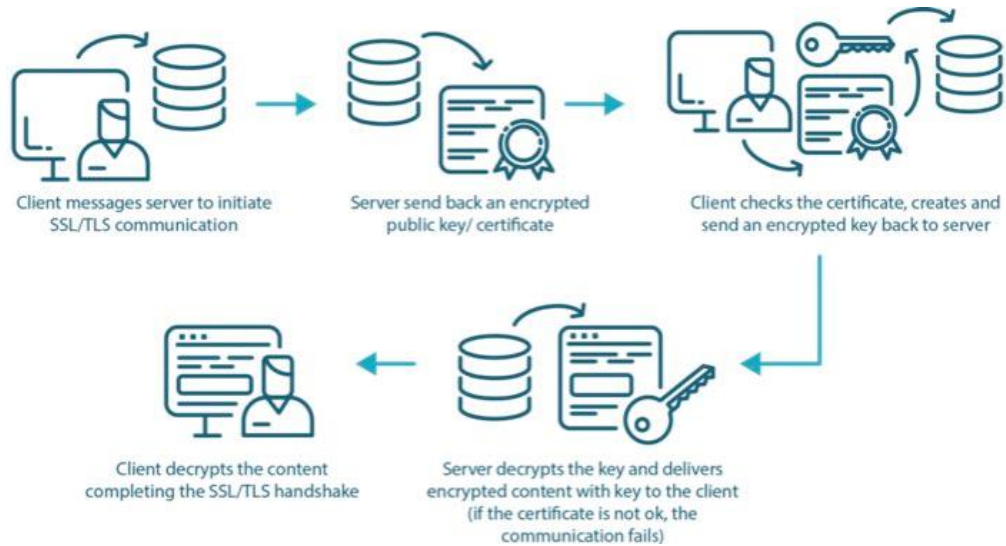
- **OpenSSL, PDO, Mbstring uzantıları:** Bu uzantılar projenin daha gelişmiş olmasına imkân tanır. php.ini dosyası üzerinden gerekli konfigürasyonlar yapılarak kullanılabilirler.

**OpenSSL:** Kriptografik kütüphanedir. OpenSSL ile web uygulamalarında SSL (Secure Sockets Layer) /TLS (Transport Layer Security) protokolü üzerinden güvenli veri alışverişi sağlanır.

SSL (Secure Sockets Layer) Güvenli Giriş Katmanı olarak Türkçeye çevrilebilir. Veri alışverişinin güvenli bir şekilde gerçekleşmesini sağlayan güvenlik protokolüdür. Veri alışverişini şifreleyerek iletişim kurduğundan gizlilik korunmaktadır. HTTPS protokolü ile kullanılır ve bu bağlantılara tarayıcılar tarafında SSL sertifikası ile erişim sağlanır, sunucuya güvenli bir şekilde erişim sağlanmış olur.

TLS (Transport Layer Security) de SSL gibi veri alışverişi güvenliğini sağlayan kriptografik bir protokoldür. İstemci ve sunucu arasında şifreli iletişim ile veri gizliliğini sağlar. TLS, SSL'in güncel versiyonu olarak da ifade edilebilir.

Açık anahtarlı kriptografi ve simetri anahtarlı kriptografi kullanır. Açık anahtarlı kriptografi ile sunucu kimliği doğrulanır ve istemci- sunucu bağlantısı şifrelenir. Simetri anahtarlı kriptografide ise veriler şifrelenir ve şifreler decrypts (şifre çözümleme) edilir.



**Şekil 7- TLS nasıl çalışır?**

**PDO (PHP Data Objects):** Veri tabanı işlemleri (veri tabanı oluşturma ve sorgulama...) gerçekleştirmek için kullanılan arayüzdür. Farklı türden veri tabanlarına (MySQL, PostgreSQL, SQLite...) erişim sağlar.

PDO sayesinde güvenli bir şekilde farklı veri tabanı sunucularında çalışılabilir ve bir hata ile karşılaşılmaması durumunda geri dönüt alınarak hata tespit edilebilir.



*Şekil 8- Lumende PDO ile erişim sağlanabilecek veri tabanı sürücüler*

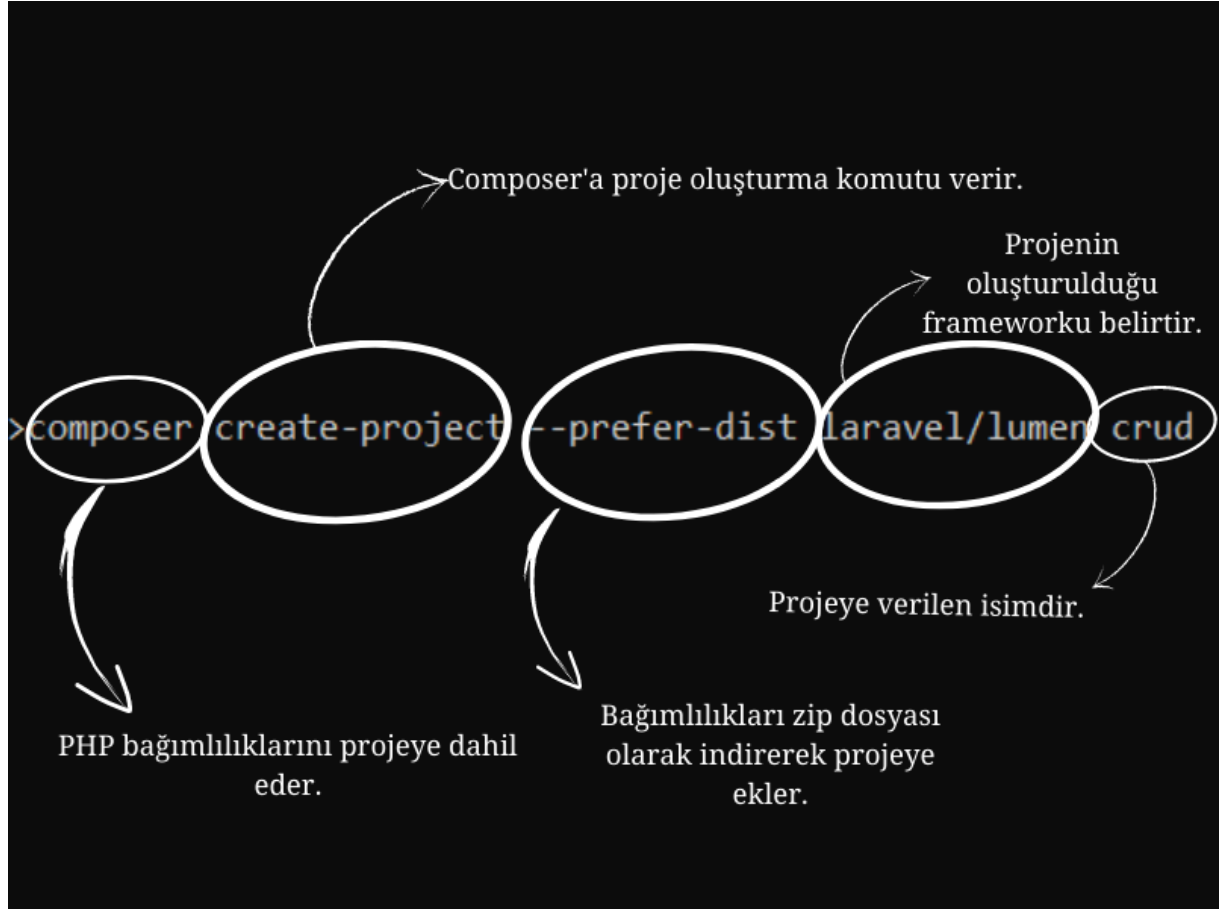
**Mbstring (Multibyte String):** Çoklu byte karakterler dizilerinin işlenmesini sağlayan php eklentisidir. Bu karakterleri doğru bir şekilde işleyebilmek ve doğru dizinleme ile dizi işlemlerinin yapılmasını sağlar. Unicode karakter kodlamaları ile kullanılır. Bu özellik php ile kurulu olarak gelir.

## 2.2. Lumen Kurulumu

Lumen ile CRUD işlemlerini (get, post, put, delete) bir REST API oluşturmak için composer üzerinde bir proje oluşturulur.

**composer create-project --prefer-dist laravel/lumen crud** komutu ile crud isimli lumen projesi oluşturulmuş olur.

Aşağıdaki görselde komutun bileşenleri yer almaktadır.



*Şekil 9-Lumen proje oluşturma komutu bileşenleri*

Verilen komut sonrasında terminal ekranında aşağıdaki ekran görüntüsünde olduğu gibi bağımlılıklar projeye dahil edilir.

```
Komut İstemi
C:\Users\hudan>composer create-project --prefer-dist laravel/lumen crud
Creating a "laravel/lumen" project at "/crud"
Info from https://repo.packagist.org: BStandwithUkraine
Installing laravel/lumen (v10.0.0)
- Installing laravel/lumen (v10.0.0): Extracting archive
Creates project in C:\Users\hudan\crud
> @php -r "file_exists('.env') || copy('.env.example', '.env');"
Loading composer repositories with package information
Updating dependencies
Lock file operations: 102 installs, 0 updates, 0 removals
- Locking brick/math (0.11.0)
- Locking doctrine/inflector (2.0.0)
- Locking doctrine/lexer (3.0.0)
- Locking dragonmantank/cron-expression (v3.3.2)
- Locking egulias/email-validator (4.0.1)
- Locking fakerphp/faker (v1.22.0)
- Locking fruitcake/php-cors (v1.2.0)
- Locking graham-campbell/result-type (v1.1.1)
- Locking guzzlehttp/uri-template (v1.0.1)
- Locking hamcrest/hamcrest-php (v2.0.1)
- Locking illuminate/auth (v10.11.0)
- Locking illuminate/broadcasting (v10.11.0)
- Locking illuminate/bus (v10.11.0)
- Locking illuminate/cache (v10.11.0)
- Locking illuminate/collections (v10.11.0)
- Locking illuminate/conditionable (v10.11.0)
- Locking illuminate/config (v10.11.0)
- Locking illuminate/console (v10.11.0)
- Locking illuminate/container (v10.11.0)
- Locking illuminate/contracts (v10.11.0)
- Locking illuminate/database (v10.11.0)
- Locking illuminate/encryption (v10.11.0)
- Locking illuminate/events (v10.11.0)
- Locking illuminate/filesystem (v10.11.0)
- Locking illuminate/hashing (v10.11.0)
- Locking illuminate/http (v10.11.0)
- Locking illuminate/log (v10.11.0)
- Locking illuminate/macroable (v10.11.0)
- Locking illuminate/pagination (v10.11.0)
- Locking illuminate/pipeline (v10.11.0)
- Locking illuminate/queue (v10.11.0)
- Locking illuminate/session (v10.11.0)
- Locking illuminate/support (v10.11.0)
- Locking illuminate/testing (v10.11.0)
- Locking illuminate/translation (v10.11.0)
- Locking illuminate/validation (v10.11.0)
- Locking illuminate/view (v10.11.0)
- Locking laravel/lumen-framework (v10.0.0)
- Locking laravel/serializable-closure (v1.3.0)
- Locking mockery/mockery (1.5.1)
- Installing symfony/polyfill-php80 (v1.28.0): Extracting archive
54 package suggestions were added by new dependencies, use "composer suggest" to see details.
Generating optimized autoload files
61 packages you are using are looking for funding.
Use the "composer fund" command to find out more!
```

**Şekil 10- Lumen proje oluşturma ve bağımlılıkların yüklenmesi**

Dosyanın oluşturulduğu dizine gidilir ve vscode üzerinden oluşturulan proje açılır.

```
C:\Users\hudan>cd crud
C:\Users\hudan\crud>code .
```

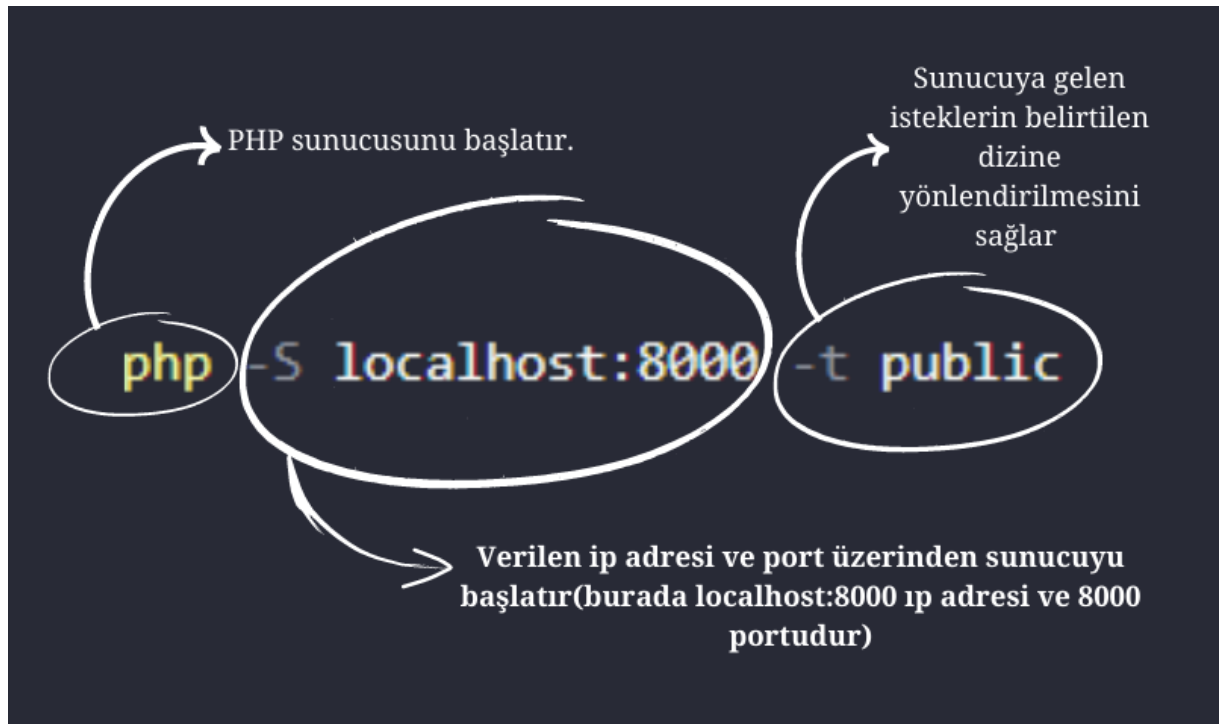
**Şekil 11- Proje VsCode ile açılır**

VsCode terminali açılarak projeyi ayağa kaldırmak için **php -S localhost:8000 -t public** komutu verilir ve proje başlatılır. Bu işlem sonrasında aşağıdaki terminal ekranı ile karşılaşırız. Bu ekran üzerinden sunucu ve istemci arasında kurulan iletişim takip edilebilir.

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE

PS C:\Users\hudan\crud> php -S localhost:8000 -t public
[Mon May 22 21:21:12 2023] PHP 8.1.0 Development Server (http://localhost:8000) started
[Mon May 22 21:21:15 2023] [::1]:59719 Accepted
[Mon May 22 21:21:17 2023] [::1]:59719 [200]: GET /
[Mon May 22 21:21:17 2023] [::1]:59719 Closing
[Mon May 22 21:21:17 2023] [::1]:59720 Accepted
[Mon May 22 21:21:17 2023] [::1]:59720 [404]: GET /favicon.ico - No such file or directory
[Mon May 22 21:21:17 2023] [::1]:59720 Closing
```

**Şekil 12- Projenin ayağa kaldırılması**



**Şekil 13- Belirtilen IP ve Portu başlatma**

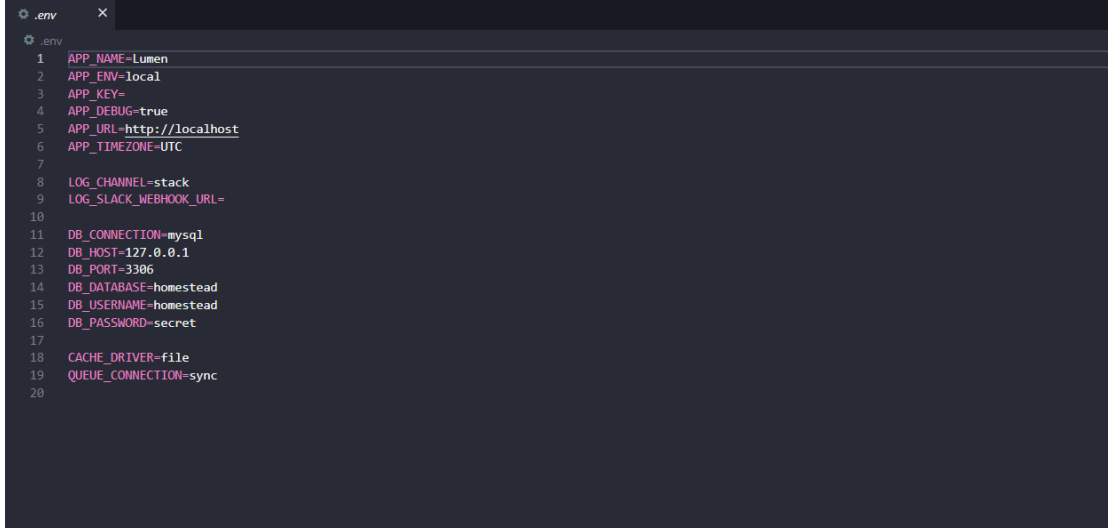
**-t public** parametresi yerel geliştirme yapılırken kullanılır. Canlı ortamda bu bileşene ihtiyaç duyulmaz.



## 2.3. Lumen Konfigürasyonları

Oluşturulan projenin doğru bir şekilde çalışması için bazı düzenlemeler yapılmalıdır.

- **.env dosyası:** Proje oluşturulduğunda varsayılan olarak gelen env dosyası şu şekildedir



```
.env
1 APP_NAME=Lumen
2 APP_ENV=local
3 APP_KEY=
4 APP_DEBUG=true
5 APP_URL=http://localhost
6 APP_TIMEZONE=UTC
7
8 LOG_CHANNEL=stack
9 LOG_SLACK_WEBHOOK_URL=
10
11 DB_CONNECTION=mysql
12 DB_HOST=127.0.0.1
13 DB_PORT=3306
14 DB_DATABASE=homestead
15 DB_USERNAME=homestead
16 DB_PASSWORD=secret
17
18 CACHE_DRIVER=file
19 QUEUE_CONNECTION=sync
20
```

Şekil 14- env dosyası

**APP** ile başlayan bölümlerde uygulamaya ait genel yapılandırmalar gerçekleştirilir. Uygulamanın adı, çalıştığı ortam, erişim anahtarı, hata ayıklama modunun kullanılıp kullanılmayacağı, projenin URL adresi ve kullanılan zaman dilimi bu alanda belirtilir. Hata ayıklama modu olan **APP\_DEBUG** yapılandırılması geliştirme aşamasında true, canlıda ise false olarak tanımlanmalıdır.

**LOG** ile başlayan kısımda ise loglama işlemlerine ait yapılandırmalar gerçekleştirilir. Yukarıda yer alan **LOG\_CHANNEL** log kayıtlarının kaydedileceği kanalın tanımlandığı alandır. Bu kanallara stack, dosya ve veri tabanı örnek verilebilir. **LOG\_SLACK\_WEBHOOK\_URL** ise tutulan logları slack üzerinden bir kanala raporlamayı sağlayan yapılandırmadır.

Yukarıda yer alan yapılandırma ayarları dışında **LOG\_LEVEL** kaydedilen log seviyesini, **LOG\_MAX\_FILES** log dosyalarının en fazla olabileceği miktarı, **LOG\_SINGLE\_CHANNEL** logları tek bir dosyada tutarak buradan takip etme işlemini, **LOG\_ENABLE\_STDERR** logların standart hata akışında (STDERR) kaydedilip edilmeyeceği ve STDERR'e yazılması durumunda hata mesajlarının STDERR'de görüntülenerek konsola yansıtılmasını, **LOG\_STDERR\_LEVEL** ise logların STDERR'e yazılma izni olduğu durum hangi log seviyesinde yazılacağını düzenler.

**DB** ile başlayan alanda veri tabanı yapılandırmaları gerçekleştirilir.

**DB\_CONNECTION** bağlantı sağlanan veri tabanı çeşidini belirtir. Oluşturulan projede varsayılan olarak mysql sürücüsü kullanılmaktadır.

**DB\_HOST** veri tabanı sunucusuna bağlanacak olan adresi (IP adresi olarak da tanımlanabilir) belirtir.

**DB\_PORT** veri tabanı sunucusunun bağlandığı port numarasıdır.

**DB\_DATABASE** projenin veri tabanının adıdır. Bağlanılacak veri tabanı ismi bu alanda yer almalıdır.

**DB\_USERNAME** veri tabanına bağlantıyı sağlarken kullanılan kullanıcı adını, **DB\_PASSWORD** ise kullanılan şifreyi belirtir.

**CACHE DRIVER:** Önbelleğe alma işleminde kullanılan sürücüyü belirtir. Bu sürücüler config klasörü altında bulunan cache.php dosyasından tanımlanır. Bu dosya altında **CACHE\_DRIVER= driver türü** olarak yapılandırılır. Tanımlanan sürücüler

- ✓ file (ön bellek dosya olarak kaydedilir)
- ✓ array (ön bellek dizi olarak tutulur)
- ✓ redis (ön bellek redis sürücüsünde tutulur)
- ✓ memcached (ön bellek redis sürücüsünde tutulur)
- ✓ database (ön bellek veri tabanına kaydedilir)

önbellek sürücüleri olabilir.

**QUEUE\_CONNECTION:** Sıralama işlemlerinde kullanılan sürücülerdir. Kuyruk işlemleri olarak da ifade edilebilir. **database, redis, beanstalkd, sqs, sync** bu sürücülere örnek olarak verilebilir. Varsayılan olarak tanımlanan sync sürücüsü işlem yönetiminin senkron (eş zamanlı) bir şekilde yönetilmesini sağlar.

Yapılandırma işlemi env dosyası üzerinden yapılır ancak bu yapılandırmalar config dizini altında da tanımlanabilmektedir.

env dosyasında yer alan yapılandırmalar sonrasında **bootstrap\app.php** dizinine gidilir. Projenin çalışması için gereken ön ayarlamalar bu dosyada yapılır.

Facade ve Eloquent bileşenleri projeye dahil edilmelidir. Facade, statik arayüz gibi davranır ve bu sayede sınıf ve nesneler kolaylıkla projeye dahil edilebilir.

Eloquent ise veri tabanı işlemlerini yönetmeyi kolaylaştırmayı sağlayan ORM (Nesne İlişkisel Eşleme) bileşenini projeye dahil etmeyi sağlar.

Bu bileşenler proje 2 şekilde dahil edilirler. İlk olarak app.php dosyasında yer alan

- ✓ **\$app->withFacades();**
- ✓ **\$app->withEloquent();**
- ✓

```
// $app->withFacades();  
  
// $app->withEloquent();
```

**Şekil 15- facade ve eloquent özelliği**

satırları yorum satırında çıkartılabilir (uncomment). Bu yöntem sayesinde \$app nesnesine facade ve eloquent özellikleri eklenmiş olur. Bu yöntem performans konusunda dezavantajlıdır.

Bir diğer yöntem ise **Dependency Injection (Bağımlılık Enjeksiyonu)** olarak adlandırılan projeye dahil edilecek bağımlılık ayarlamalarını direkt olarak dahil etme işlemidir. **app()** metodu direkt kullanılarak bağımlılıklar projeye eklenir ve bu sayede performansta herhangi bir kayıp meydana gelmez.

Aşağıdaki ekran alıntısında app() metodu ile facade ve eloquent sınıfları projeye dahil edilmiştir.

```
use Illuminate\Database\Capsule\Manager as Capsule;  
use Illuminate\Support\Facades\Cache;  
  
//facade sınıfını dahil etme  
$value = Cache::get('key');  
  
//eloquent sınıfını dahil etme  
$capsule = app(Capsule::class);  
$capsule->bootEloquent();
```

**Şekil 16- facade ve eloquent sınıfları app() metodu ile projeye dahil etme**

Cache sınıfı facade olarak kullanılarak ön bellekte bulunan **key** değeri alınır.

Capsule sınıfı ise projeye enjekte edilerek bootEloquent() yöntemini çağırır, böylece veri tabanı işlemlerini yönetecek olan sınıf projeye dahil edilmiş olur.

### 3. CRUD İşlemlerini Gerçekleştiren REST API Oluşturma

#### 3.1. CRUD İşlemleri Nelerdir?

CRUD işlemleri veri tabanı üzerinde yapılan işlemlerdir. Create, Read, Update ve Delete işlemleri olarak açıklanır. Bu işlem veri tabanı üzerinde en sık kullanılan işlemlerdir.

**Create** (Oluşturma): Veri tabanına veri ekleme işlemidir.

**Read** (Okuma): Veri tabanında yer alan kayıtları okuma işlemidir.

**Update** (Güncelleme): Veri tabanında kayıtlı olan veri üzerinde değişiklik yapma işlemidir.

**Delete** (Silme): Veri tabanında kayıtlı olan veriyi kalıcı olarak silme işlemidir.

#### 3.2. Route Tanımlamaları

Route'lar, url yönlendirmesini sağlar ve endpoint noktalarının tanımlandığı yerdir. **routes/web.php** klasörü altında oluşturulurlar.

Kullanıcı işlemlerini gerçekleştirmek ihtiyacımız olan routeleri tanımlayalım.

```
$router->group(['prefix' => 'api'], function () use ($router) {  
    $router->get('index', 'UserController@index');  
    $router->post('store', 'UserController@store');  
    $router->get('show/{id}', 'UserController@show');  
    $router->put('update/{id}', 'UserController@update');  
    $router->delete('delete/{id}', 'UserController@delete');  
});
```

*Şekil 17- route tanımlamaları*

Yukarıda kullanıcı işlemlerinin gerçekleşmesi için gerekli routing (yönlendirmeler) yapılmıştır. Tanımlanan routelar gruplandırılarak **/api** uzantısı ile başlayacak şekilde ayarlanmıştır.

UserController içerisinde yer alan metotlara göre yönlendirilen url adreslerini inceleyelim.

Yapılan istek türleri 2 parametreye sahiptir. Bu parametrelerden ilki endpoint noktalarıdır. Tanımlanan url ve prefix (ön ek) sonrasında hangi endpoint ile çağırılacağını tanımlandığı alandır. İkinci parametre ise isteği yapacak olan kaynağı belirtir. Yani kullanılacak controller ve burada yer alan metot üzerinden istek gerçekleştirilir.

get metodunun kullanıldığı index ve show metotları görüntüleme işlemlerini sağlamak için url'ye get isteği atar. Bu isteğin doğru olması durumunda dönen response, istek verilerini görüntüler. Show metodunda bu görüntüleme işlemi idye göre yapılmaktadır.

url adresine atılan post isteği, UserController dosyasında bulunan store fonksiyonu işlemlerini kullanarak veri ya da kaynak ekleme işlemi gerçekleştirir.

put isteği ile kayıtlı olan bir veriyi güncellemek üzere url adresine istek gönderilir. Bu istek UserController'da bulunan update metodunu kullanarak sağlanır. Parametre olarak verinin idsini kullanır.

delete isteği, kayıtlı verinin kalıcı olarak veri tabanından silinmesi için kullanılır ve id parametresi alır.

Tanımlanmış route'lar bu işlemlerin gerçekleşmesi için yönlendirme sağlarlar.

### 3.3. Migration Oluşturma

Migration işlemleri, veri tabanı tabloları oluşturma ve düzenleme olaylarıdır. Migrationlar veri tabanı yönteminde kullanılırlar. Bir Lumen projesine Migration eklemek için Laravel özelliklerinden faydalanılması gerekir. Lumende direkt olarak gerçekleştiremediğimiz için lumen-generator yardımı ile bu işlemi kolaylıkla gerçekleştirebiliriz.

Lumen generator, **composer require flipbox/lumen-generator** komutu ile projeye yüklenerek mevcut laravel özelliklerini projede kullanmaya imkân tanır. Bu komut çalıştırılmadan önce **Bootstrap/app.php** dosyası içerisine

```
$app->register(Flipbox\LumenGenerator\LumenGeneratorServiceProvider::class);
```

eklenerek projeye Lumen Generator servis sağlayıcısı yüklenir. Daha fazla bilgi için github adresinden yararlanabilirsiniz.

<https://github.com/flipboxstudio/lumen-generator>

Sonrasında migration oluşturmak için kullanılan Laravelde kullanılan

```
PS C:\Users\hudan\crud> php artisan make:migration user
```

database/migrations klasörü altında user dosyası oluşturulur. Oluşturulan user tablosu şu şekildedir

```

database > migrations > 2023_05_23_132304_user.php > ...
1  <?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  return new class extends Migration
8  {
9      /**
10       * Run the migrations.
11       */
12       public function up(): void
13       {
14           //
15       }
16
17       /**
18       * Reverse the migrations.
19       */
20       public function down(): void
21       {
22           //
23       }
24  };
25

```

**Şekil 18-varsayılan user migration dosyası**

Dosya içerisindeki up() metodu, oluşturulan tablo üzerinde yapılacak işlemler içindir. Tablo içerisinde sütun tanımlamaları ve işlemleri burada yapılır.

down() metodu ise tabloda yapılan değişiklikleri geri almak için kullanılır.

```

env app.php web.php 2023_05_23_132304_user.php
database > migrations > 2023_05_23_132304_user.php > ...
1  <?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  return new class extends Migration
8  {
9      /**
10       * Run the migrations.
11       */
12       public function up(): void
13       {
14           Schema::create('users', function (Blueprint $table) {
15               $table->increments('id');
16               $table->string('name');
17               $table->string('surname');
18               $table->string('email')->unique();
19               $table->string('password');
20               $table->timestamps();
21           });
22       }
23
24       public function down(): void
25       {
26           Schema::drop('users', function (Blueprint $table) {
27               Schema::dropIfExists('users');
28           });
29       }
30  };
31
32

```

**Şekil 18-tablo oluşturma ve sütun olayları**

Yukarıda oluşturulan users tablosu ve içerisinde bulunacak olan sütunlar özellikleri ile tanımlanmaktadır.

```
PS C:\Users\hudan\crud> php artisan migrate

INFO Preparing database.

Creating migration table ..... 14ms DONE

INFO Running migrations.

2023_05_23_132304_user ..... 29ms DONE
```

*Şekil 19-veri tabanında tablo oluşturma komutu*

**php artisan migrate** komutu ile user dosyasında bulunan fonksiyonlar çalışarak tablo veri tabanında oluşturmuş olur.

### 3.4. Controller Dosyası Oluşturma

Controller dosyaları model ve view arasında iletişimi sağlayan birimdir. Route dosyasında tanımladığımız isteklerin içerikleri bu dosyada yer alır. Burada tanımlanan metotlar, web adresine gönderilen istekler sonucu çalışır ve bir cevap(response) oluştururlar.

Şimdi projedeki istekleri gerçekleştirecek controller dosyasını oluşturalım.

```
PS C:\Users\hudan\crud> php artisan make:controller UserController
Controller created successfully.
```

*Şekil 20-Controller dosyası oluşturma*

Oluşturulan controller dosyasında kullanıcı işlemleri gerçekleştirilecektir. Routelerini belirlediğimiz fonksiyonlar bu dosyada oluşturulur. Bunlardan ilki **index ()**'tir. Burada kayıtlı olan tüm verilerin listelenmelidir. Bu sebeple User modelinde bulunan tüm veriler index fonksiyonu tarafından döndürülür.

```
public function index()
{
    return User::all();
}
```

*Şekil 21-index fonksiyonu*

Sonraki metot ise store metodudur. Bu metot kaynağa post isteği gönderir, yeni veri kaydetmek için kullanılır.

```
public function store(Request $request)
{
    $validate = $this->validate($request, [
        'email' => ['required', 'email', 'max:255'],
        'password' => ['required', 'min:8'],
    ]);

    $user= new User();
    $user->name=$request->name;
    $user->surname=$request->surname;
    $user->email=$request->email;
    $user->password=Hash::make($request->password);
    $user->save();
    return response()->json($user);
}
```

**Şekil 22-store fonksiyonu**

show() metodu, sunucuya get isteği göndererek idsi parametre olarak verilen veriyi User modelinde bulur ve ekranda görüntülenmesini sağlar.

```
public function show($id)
{
    $user= User::find($id);
    return response()->json($user);
}
```

**Şekil 23-show fonksiyonu**

update() metodu, kaynağa put isteği gönderir, veri güncellemek için kullanılır. id parametresini alarak güncellenecek kullanıcıyı bulur ve güncelleme işlemini gerçekleştirir. put isteği ile güncelleme yapabilmek için güncelleme yapılacak veriye ait tüm veriler parametre olarak verilmelidir. Fonksiyonun içeriği aşağıdaki şekildedir

```
public function update($id,Request $request)
{
    $user= User::find($id);
    $user->name=$request->name;
    $user->surname=$request->surname;
    $user->email=$request->email;
    $user->password=Hash::make($request->password);
    $user->save();
    return response()->json($user);
}
```

**Şekil 24-update fonksiyonu**



delete() metodu, id parametresi ile silinmek istenen veriyi bularak veri tabanından kalıcı olarak siler. Bu işlem delete isteği kullanılarak gerçekleştirilir.

```
public function delete($id)
{
    $user = User::find($id);
    $user->delete();
    return response()->json('User deleted');
}
```

**Şekil 25-delete fonksiyonu**

Aşağıda UserController dosyasının tamamı görüntülenmektedir.

```
app > Http > Controllers > UserController.php > UserController > update
3 namespace App\Http\Controllers;
4 use Illuminate\Http\Request;
5 use App\Models\User;
6 use Illuminate\Support\Facades\Hash;
7 class UserController extends Controller
8 {
9     public function index()
10     {
11         return User::all();
12     }
13     public function store(Request $request)
14     {
15         $validate = $this->validate($request, [
16             'email' => ['required', 'email', 'max:255'],
17             'password' => ['required', 'min:8'],
18         ]);
19         $user= new User();
20         $user->name=$request->name;
21         $user->surname=$request->surname;
22         $user->email=$request->email;
23         $user->password=Hash::make($request->password);
24         $user->save();
25         return response()->json($user);
26     }
27     public function show($id)
28     {
29         $user= User::find($id);
30         return response()->json($user);
31     }
32     public function update($id,Request $request)
33     {
34         $user= User::find($id);
35         $user->name=$request->name;
36         $user->surname=$request->surname;
37         $user->email=$request->email;
38         $user->password=Hash::make($request->password);
39         $user->save();
40         return response()->json($user);
41     }
42     public function delete($id)
43     {
44         $user = User::find($id);
45         $user->delete();
46         return response()->json('User deleted');
47     }
48 }
```

**Şekil 21-Controller dosya içeriği**

### 3.5. Seeder Kullanımı

Oluşturduğumuz proje üzerinde geliştirmeler yapmak için bir veri grubuna ihtiyaç duyarız. Bu veri grubunu seeder sınıfı ile oluşturabiliriz. Bu sayede tek tek veri eklemek yerine otomatik bir şekilde, istenilen miktarda veriyi veri tabanına kaydedebiliriz.

**database->factories->UserFactory.php** dosyasına gidilir, bu dosyada bulunan **definition()** metodu ile istenilen verilerin nasıl oluşturulacağı tanımlanır. Aşağıdaki ekran alıntısında faker kütüphanesi kullanarak rastgele veriler üretilmesi istenmektedir.

```
database > factories > UserFactory.php > ...
1  <?php
2
3  namespace Database\Factories;
4
5  use App\Models\User;
6  use Illuminate\Database\Eloquent\Factories\Factory;
7  use Illuminate\Support\Facades\Hash;
8
9  class UserFactory extends Factory
10 {
11     /**
12      * The name of the factory's corresponding model.
13      *
14      * @var string
15      */
16     protected $model = User::class;
17
18     /**
19      * Define the model's default state.
20      *
21      * @return array
22      */
23     public function definition()
24     {
25         return [
26             'name' => $this->faker->name(),
27             'surname' => $this->faker->lastName(),
28             'email' => $this->faker->unique()->safeEmail(),
29             'password' => Hash::make('password'),
30         ];
31     }
32 }
```

**Şekil 22-oluşturulacak veri tanımlanması**

Bu verileri üretebilmek için bir seeder oluşturulmalıdır. **php artisan make:seed UsersTableSeeder** komutu kullanılarak **UsersTableSeeder** isimli seeder dosyası oluşturulur.

```
PS C:\Users\hudan\crud> php artisan make:seed UsersTableSeeder
INFO Seeder [C:\Users\hudan\crud\database\seeders\UsersTableSeeder.php] created successfully.
```

Oluşturulan dosyaya gidilerek run metoduna kullanıcı oluşturmak için gereken kod satırları yazılır.

```
public function run(): void
{
    User::factory()->count(10)->create();
}
```

**Şekil 23-rastgele veri oluşturma**

Run metodunda bulunan kod, User modeli ile UserFactory sınıfını kullanarak 10 adet rastgele veri oluşturmayı ve bunları veri tabanına eklemeyi sağlar.

Tanımlanan ifadeleri çalıştırmak ve böylece verileri oluşturarak veri tabanına eklemek için **php artisan db:seed** komutu kullanılır.

```
PS C:\Users\hudan\crud> php artisan db:seed --class=UsersTableSeeder  
  
INFO Seeding database.
```

**Şekil 24-UserTableSeeder sınıfını çalıştırma**

Oluşan veriler users tablosuna kaydedilir ve aşağıdaki gibi bir veri grubu oluşur.

Seenekler

1

Düzenle

Kopyala

Sil

1

Mr. Nat Medhurst

Lebsack

pballstrier@example.com

\$2y\$10\$qpinq9fY.OuwtS4IBTujqGrgPETdd23jxusyzDGC...

2023-05-23 22:11:21

2023-05-23 22:11:21

2

Düzenle

Kopyala

Sil

2

Prof. Furman Auer

Brown

theodora.harris@example.net

\$2y\$10\$shKNWxsBwl1xZk7JT6fUaerch/XsZVHCLQpobdIT...

2023-05-23 22:11:21

2023-05-23 22:11:21

3

Düzenle

Kopyala

Sil

3

Mrs. Bria Satterfield

Lowe

nbarrows@example.net

\$2y\$10\$mPRYNBCBAQcpYL4gSKYhl.J6zNCrIRmj/pkKuNNWaVyi...

2023-05-23 22:11:21

2023-05-23 22:11:21

4

Düzenle

Kopyala

Sil

4

Prof. Jane Bartoletti Sr.

Murazik

constantin.schneider@example.net

\$2y\$10\$X/Vwe/R3slhEW7XuuEqnyOx2T7xdlwANM0rOFmNTd1b...

2023-05-23 22:11:21

2023-05-23 22:11:21

5

Düzenle

Kopyala

Sil

5

Vivienne Kertzmman

Goodwin

qtrantow@example.org

\$2y\$10\$0KH.9nULDWCx2WMWAIzNerd7bOh1oKbS4.KQmrf92m...

2023-05-23 22:11:21

2023-05-23 22:11:21

6

Düzenle

Kopyala

Sil

6

Dr. Derick Gusikowski Sr.

Hintz

wrogahn@example.org

\$2y\$10\$ide3yLnBGGrJtzc6RNhmiu8J2GpGCTyqxGA0pVpchLI...

2023-05-23 22:11:21

2023-05-23 22:11:21

7

Düzenle

Kopyala

Sil

7

Dr. Quentin Stokes Sr.

Cummerata

hkris@example.com

\$2y\$10\$yh8GpjsbWd5D5z3th/7uJnCrQPegNGcT9N9IOdlGB...

2023-05-23 22:11:21

2023-05-23 22:11:21

8

Düzenle

Kopyala

Sil

8

Torey Schinner

Schaefer

mueller.tomas@example.org

\$2y\$10\$m6Zlum6xhsu RHu5V6Fye.cYZVCi9p0aGAMOA83nZNF...

2023-05-23 22:11:21

2023-05-23 22:11:21

9

Düzenle

Kopyala

Sil

9

Lorenzo Smith

Larkin

mlarson@example.org

\$2y\$10\$itMZIn4ere4UQ1ZPKHxto.chYIEJd3DB6c5LU/nl/3W...

2023-05-23 22:11:21

2023-05-23 22:11:21

10

Düzenle

Kopyala

Sil

10

Ms. Juliana Goldner

Connelly

hermiston.agnes@example.org

\$2y\$10\$4bwquqKMqYQ4CSFtaMm3au1DSqx2Y7/.6lqT1skJmd...

2023-05-23 22:11:21

2023-05-23 22:11:21

Tümüü İşaretle

Seilileri:

Düzenle

Kopyala

Sil

Dışa aktar

**Şekil 25-users tablosuna eklenen veriler**

## 4. API Test İşlemleri

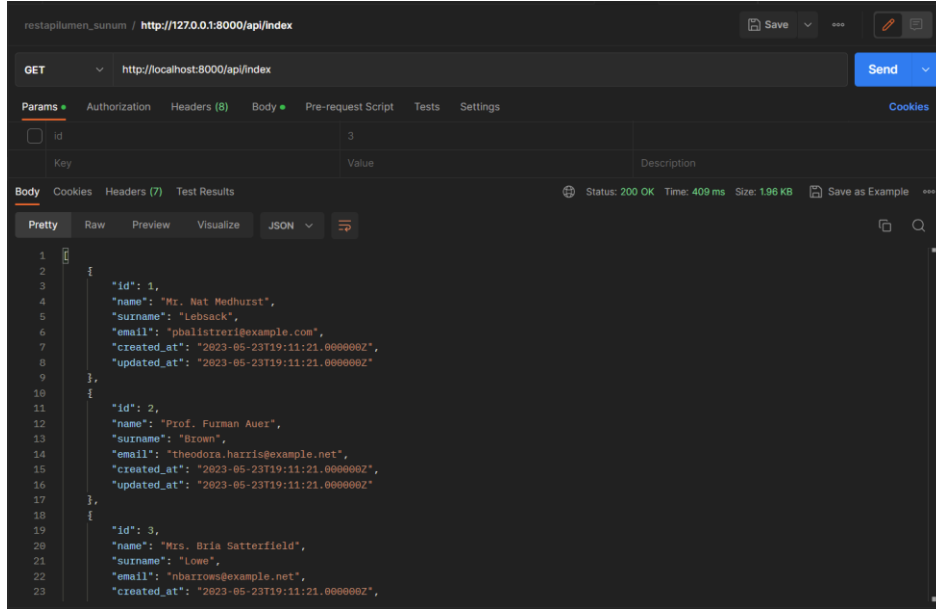
### 4.1. Postman Nedir?

Proje geliştirirken kullanılan API test etme aracıdır. Postman üzerinden isteklerde bulunarak oluşturulan API kontrol edilebilir.

### 4.2. Postman ile API Testleri

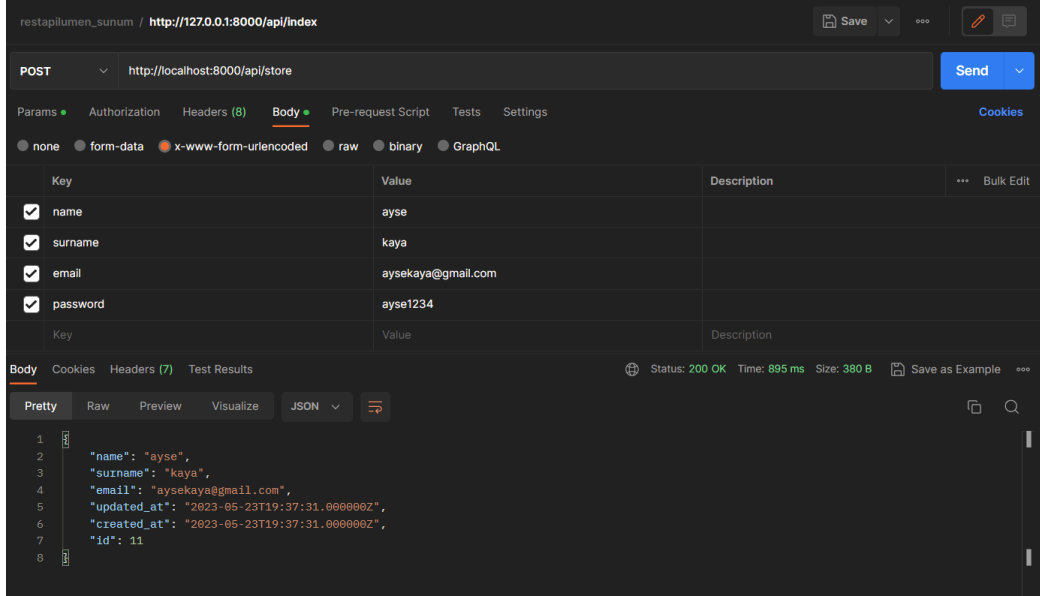
Oluşturduğumuz proje için gerekli API testlerini postman üzerinden kontrol edelim.

İlk olarak get isteği göndererek index metodu ile tüm User verilerini listeleyelim.



Şekil 26- get isteği ile verileri listeleme

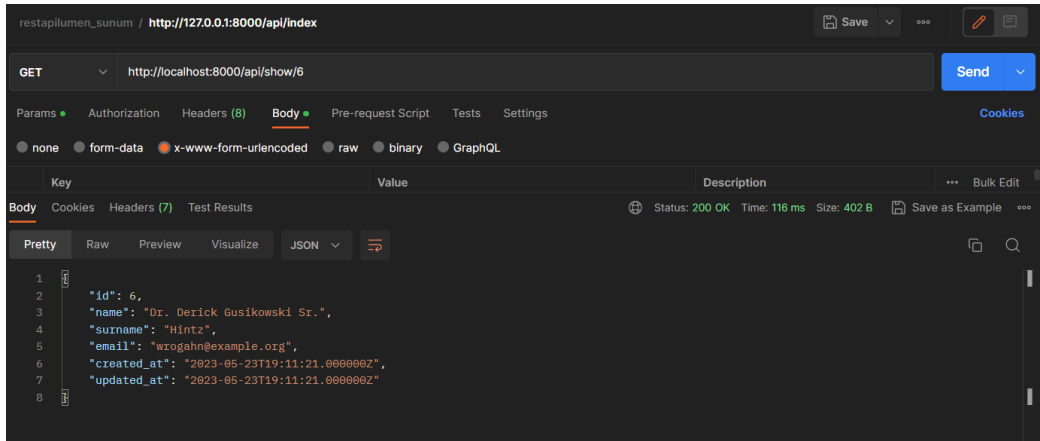
post isteđi ile User modelinde bir veri ekleyerek veri tabanına kaydedelim.



**Şekil 26-post isteđi ile veri kaydetme**

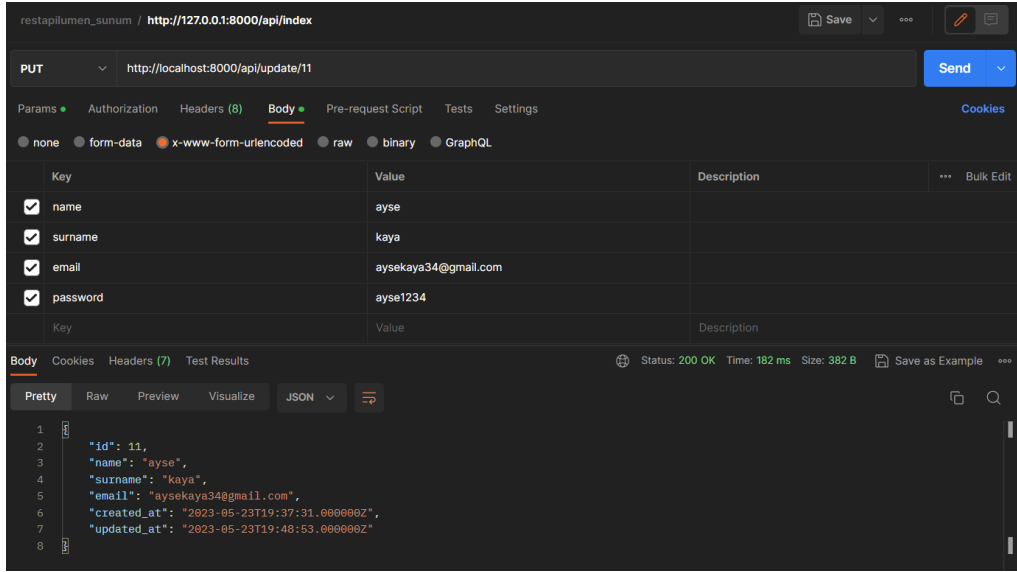
Yukarıdaki ekranda anlaşılacağı üzere istek sonucunda kaydedilen veri, json formatında bir response çıktı olarak döndürülmüştür.

show metodu ile id parametresini alarak get isteđi yapılan veriyi postman üzerinden görüntüleyelim.



**Şekil 27-get isteđi ile istenilen veriyi görüntüleme**

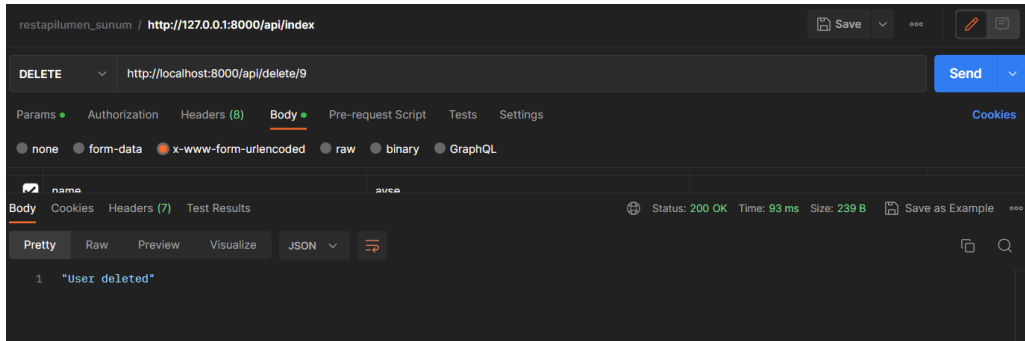
update metodu ile web adresine put isteđi göndererek id ve kayıtlı kullanıcı bilgileri kullanılarak güncelleme işlemi gerçekleştirilim.



**Şekil 28-put isteđi ile veri güncelleme**

yukarıda kullanıcı emaili güncellenerek veri tabanına kaydedilmiştir.

Url adresine delete isteđi göndererek, delete metodu ile kayıtlı olan veriyi kalıcı olarak silelim.



**Şekil 29-delete isteđi ile veri silme**

## KAYNAKÇA

- <https://www.geeksforgeeks.org/what-are-the-differences-and-similarities-between-lumen-and-laravel/>
- <https://lumen.laravel.com/docs/10.x>
- <https://github.com/laravel/lumen>
- <https://auth0.com/blog/developing-restful-apis-with-lumen-a-php-micro-framework/>
- <https://chat.openai.com/>
- <https://www.hosting.com.tr/bilgi-bankasi/rest-api/>
- <https://www.codecademy.com/article/what-is-rest>
- <https://zeo.org/tr/kaynaklar/blog/http-durum-kodlari-rehberi>
- <https://tech.bodyfitstation.com/development/web-development/php/lumen/learning/download-and-install-lumen/>
- <https://cdn.hosting.com.tr/blog/wp-content/uploads/2022/12/tls-nasil-calisir.jpeg>
- <https://www.hosting.com.tr/blog/tls/>
- <http://www.yucelalkan.com/pdo-nedir>
- <https://github.com/flipboxstudio/lumen-generator>