İSTANBUL TEKNİK ÜNİVERSİTESİ
Bilgisayar ve Bilişim Fakültesi

PHP ile JSON Web Servis Entegrasyonu ve Laravel Framework

STAJ

Bünyamin İbrahim KURT
150140145

YAZ / 2017

İstanbul Teknik Üniversitesi

Bilgisayar ve Bilişim Fakültesi

STAJ RAPORU

Akademik Yıl: 2016 - 2017

Staj yapılan dönem: Yaz

Öğrenci ile ilgili bilgiler

Adı ve Soyadı:	Bünyamin İbrahim KURT
Öğrenci Numarası:	150140145
Bölüm:	Bilgisayar Mühendisliği
Program:	%100 İngilizce
E-posta Adresi:	kurtbu@itu.edu.tr
(Cep) Tel No:	0531 851 0467
ÇAP öğrencisi misiniz?	Hayır
Mezuniyet durumunda mısınız?	Hayır
Yaz okulunda ders alıyor musunuz?	Hayır

Öğrencinin çalıştığı kurum ile ilgili bilgiler

İsmi:	Digital Trade Yazılım Bilişim Eği. Dan. Ve Turizm Tic. A.Ş.
Birimi:	Yazılım
Web Adresi:	www.digitaltrade.com.tr
Kısa Adresi:	İstanbul Üniversitesi Avcılar Kampüsü Entertech Teknokent Binası 3.Kat Ofis: 322 Avcılar / İstanbul

Yetkili kişi ile ilgili bilgiler

Bölümü:	Yazılım
Unvanı:	IT Drektör
Adı ve Soyadı:	Necdet Tenekeci
(Kurumsal) E-posta:	necdettenekeci@digitaltrade.com.tr
(Kurumsal) Tel. No.:	+90 212 691 64 26

Yapılan iş ile ilgili bilgiler

Staj yeri	İstanbul Üniversitesi Avcılar Kampüsü Entertech Teknokent Binası 3.Kat Ofis:322 Avcılar/İstanbul
Staj başlangıç tarihi	03.07.2017
Staj bitiş tarihi	28.07.2017
Stajda çalışılan net **gün** sayısı	20
Staj süresince sigortanız var mıydı?	Evet, İTÜ tarafından sigortalandım.

STAJ RAPORU ONAY FORMU

1 İÇİNDEKİLER

150140145 numaralı, Bünyamin İbrahim Kurt adlı öğrencinin, yukarıda "İçindekiler" bilgisi verilen staj raporu, görülmüş ve uygun bulunmuştur.

Formu Dolduran Firma Yetkilisinin Adı ve Soyadı: Necdet Tenekeci

Yetkilinin Unvanı: IT Direktör

Müdür-İmza-Kaşe:

(Kurumsal) E-posta: necdettenekeci@digitaltrade.com.tr

(Kurumsal) Tel. No: 0212 691 64 26

2 KURUMLA İLGİLİ BİLGİ

Digital Trade A.Ş. İstanbul Üniversitesi Avcılar Kampüsü Entertech Teknokent binasında 2015 yılında kurulan e-ticaret odaklı bir teknoloji firmasıdır. Digital Trade bünyesinde bulundurduğu Bilet.com ve Elbise.com gibi markalarının yazılımını kendi bünyesi içinde geliştirmektedir. Firma özellikle Bilet.com markasını büyütmek adına yeni entegrasyon süreçlerine dahil olmaktadır. Bende staj sürem boyunca bu sürecin içine dahil oldum. Firma kendi içinde IT, pazarlama ve müşteri hizmetleri şeklinde 3 ana bölüm ve örgütlenmeden oluşmaktadır. Firmanın uzmanlık alana yazılım olmakla beraber danışmanlık hizmeti de verebilmektedir. Firma misyonunu şu şekilde açıklamaktadır: "Faklı sektörlerde pazar potansiyeli olan inovatif projeleri keşfedip yenilikçi teknoloji çözümleri ile ürünleştirmek ve ticarileştirmek."

Digital Trade çalışmalarını Bilet.com markasını büyütmek için düzenlemketedir. Şuanda Bilet.com markasında Uçak, Otobüs, İDO, BUDO ve Yunan adalarının biletleri satılmaktadır. Benim görev yaptığım kısım Yunan adaları biletlerinin entegrasyon sürecine yardımcı olmaktı. Firma bu geliştirmeleri kendi bünyesinde barındırdığı yazılımcı çalışanlar ile yapmaktadır.

Şirketin önem verdiği değerler başlıca şunlardır:

- Müşteri odaklı çalışma.
- Hızlı çözümler üretebilmek.
- İnovatif çözümler geliştirmek.
- Yeni teknolojilerden faydalanma
- Çevik yönetim

3 GİRİŞ

2017 yaz dönemi boyunca ikinci stajımı Digital Trade A.Ş. şirketinde yapmış bulunmaktayım. 20 gün boyunca İstanbul Üniversitesi Teknoparkta çalışma imkanı buldum. Teknoparklarda şirketlerin nasıl çalışma olanakları olduğunu ve şirketlerin çalışma ortamlarını inceleme fırsatım oldu.

Staj boyunca daha önceden deneyimlemiş olduğum teknolojileri kullanıp kendimi geliştirme fırsatı yakaladım. Stajımı yaptığım şirket geliştirmelerinin PHP dili ile yapmaktaydı. Benimde stajımın konusu ve amacı PHP dilini kullanarak firmanın anlaşmalı olduğu bir feribot şirketinin sağladığı JSON formatındaki web servisin entegrasyon sürecine dahil olmak ve yardımcı olmaktı. Ayrıca, şirketin kendi içinde raporlamaları takip ettiği panelin düzenlenmesinde rol aldım. Panel ön yüzde bootsrap, arka tarafda Laravel frameworku ile geliştirilmekte olup, bu süreç boyunca MVC yapısını öğrenme imkanı buldum. Genel olarak stajım boyunca bu iki iş üzerinde çalıştım.

4 STAJ PROJESİNİN TANIMI ve ANALİZİ

Staj sürecim boyunca yukarıda da belirttiğim gibi bir feribot web servisinin entegrasyonunda görev aldım. Web servis için PHP ile request ve response sınıflarının hazırlanmasını sağladım. Ayrıca satışların ve raporların takip edileceği panelin düzenlenmesinde yine görev aldım.

4.1 PHP ile JSON Formatındaki Web Servis Entegrasyonu

Öncelikle yaptığımız projeyi detaylı olarak anlatmak istiyorum. Projemiz şirketin bünyesinde barındırdığı Bilet.com sitesine yeni bir ürün eklemekti. Bu ürün firmanın anlaştığı tedarikçi bir firma üzerinden bize sağlanan yunan adalarına giden feribot biletlerinin web servis ile entegre edilip siteye eklenmesiydi. Halihazırda Bilet.com sitesinde İDO ve BUDO firmalarının feribot bilet satışı daha önceden entegrasyon yapıldığı için bulunmaktaydı. Bizim de amacımız bunlara bir yenisini eklemek olacaktı. Benim ana görevim daha önceden İDO ve BUDO için yazılmış olan kodları yeni web servis için uygun hale getirmekti. Request ve response sınıflarının en baştan yazılması, exception sınıflarının düzenlenmesi vb. gibi görevleri bu staj süreci boyunca üstlendim. Tabi özellikle request ve response sınıflarını yazarken şirketin anlaşmış olduğu firmadan bize iletilen web servis dokumanından faydalanarak bu süreci gerçekletirdim. Bu süre boyunca diğer firmadan sorularımız olduğu zaman bize destek verebilecek bir kişi ile iletişim halindeydik. Bu sayede serviste takıldığımız veya anlamadığımız yerleri sorma imkanına sahiptik.

4.1.1 Kullandığım Teknolojiler

Bu projede kullandığım teknolojiler kısa açıklamaları ile şunlardır:

- **PHP**: Web siteleri ve web uygulamaları geliştirilmek için oluşturulmuş web tabanlı çalışan bir programlama dilidir.
- **Netbeans**: Oracle tarafından geliştirilmiş temel olarak Java için tasarlanmış olan bir geliştirme platformudur. Bende geliştirmelerimi bu platform üzerinde gerçekleştirdim.
- **JSON**: Kısaca bir çeşit veri formatıdır. Avantajı XML'den daha az yer kaplamasıdır. Benim de entegre ettiğim web servisi JSON formatında veri kabul edip, yine JSON formatında veri döndürüyordu.

4.1.2 Kodlar ve Açıklamaları

Öncelikle ilk yaptığımız iş webservisten nereden ve nereye olan istekleri yapıp bu limanların ve iskelelerin IDlerin kaydetmekti. Biz projemizi yaparken bazı istekleri kolay hızlı olması açışından Postman uygulamasını kullandık. Bu sayede istediğimiz zaman istediğimiz metodu çağırıp sonuçlarını kontrol edebiliyorduk. Ayrıca webservisten istek yapmadan önce headera yerleştirmek üzere bir defalık bir token isteği yaptık. Almış olduğumuz bu tokenı tüm isteklerin headerına ekleyip isteklerimizi bu şekilde gerçekleştirdik.

Daha sonra PHP de istekleri yaptığımız daha önceden diğer webservisler için yazılmış olan Client sınıfını yeniden oluşturduk. Yapılan tüm istekler buradan yapılmaktadır. İsteklerimizi PHPnin Guzzle kütüphanesi ekleyerek guzzle üzerinden gerçekleştirdik.

```php
public function send(RequestInterface $request) {

    $params = $request->getParams();
    $endpoint = $request->getEndPoint();
    $class = get_class($request);

    $payload = json_encode($params);

    try {

        $http_request = new Request('POST', $this->config['endpoint'] . $endpoint, [], $payload);

        $response = $this->Client->send($http_request);

        $response_string = $response->getBody()->getContents();

        $response_object = json_decode($response_string);

        //
    } catch (RequestException $ex) {

        $this->Logger->log($class, $endpoint, $payload, null, $ex);

        $response = $ex->getResponse();

        if ($response !== null && $response->getStatusCode() == 401) {

            // Remove old token from cache. Next request will get new token.
            Cache::forget('erturk_token');

            throw new TokenException($ex->getMessage());
        }

        throw new HttpErrorException($ex);
        //
```

Şekil 1: Client sınıfının içindeki send metodu.

Client sınıfının send metodunda temel olarak istek (request) sınıfından gelen parametleri endpoint POST metodu ile çağırıyor. Gelen cevabı da geriye döndürüyor.

```php
public function __construct(HttpLoggerInterface $Loger, array $config) {

    $this->Logger = $Loger;
    $this->config = $config;

    $token = $this->getToken();

    $this->Client = new Guzzle([
        'timeout' => 20,
        'verify' => false,
        'headers' => [
            'Authorization' => isset($token) ? 'bearer ' . $token : 'bearer ' . $config['token'],
            'Content-Type' => 'application/json',
            'Accept' => 'application/json'
        ]
    ]);
}
```

Şekil 2: Client sınıfının constructor

Client sınıfının constructorunda config dosyasında kayıtlı olan tokenı oluşturulan isteğin headerına eklenmesi yapılıyor.

Client sınıfını tamamladıktan sonra request ve response sınıflarını hazırladım. Request sınıflarında webservise uygun olarak parametlerin uyarlanması gerekiyordu. Bende webservis dokumantasyonuna göre bu ayarlamaları yaptım. Aşağıda örnek olarak getRoutesRequest ve getRoutesResponse sınıflarını açıklayacağım. getRoutesRequest metodu iki iskele arasındaki seferleri döndüren temel isteklerden bir tanesidir. Parametre olarak iskele IDleri, gidiş ve isteğe bağlı olarak dönüş tarihi ve yolcu bilgileri gibi bilgileri gönderiyoruz. Cevap olarak bize seferler JSON dizisi olarak geri dönüyor.

```php
protected $endpoint = 'api/booking/ticket/search';
```

Şekil 3: getRoutesRequets sınıfının endpoint değişkeni

getRequest metodunda ilk yaptığım iş Client sınıfında isteğin hangi URL'e yapılacağının bilinmesi için endpoint değişkenini tanımladım.

```php
protected function setParams() {

    // We will use this data in Response object.
    $this->responseData = [
        'passengers' => $this->data['passengers'],
        'date' => $this->data['date'],
        'originID' => $this->Converter->portCodeToID($this->data['origin']),
        'arrivalID' => $this->Converter->portCodeToID($this->data['destination']),
        'vehicle' => [
            'count' => $this->data['vehicle']['count'],
            'code' => $this->data['vehicle']['code']
        ]
    ];

    $this->params = [
        'LangCode' => 'TR',
        'RouteType' => 'OneWay',
        'ServiceDate' => $this->data['date'],
        'DepartureRegionID' => $this->Converter->portCodeToID($this->data['origin']),
        'ArrivalRegionID' => $this->Converter->portCodeToID($this->data['destination']),
        'AdultQuantity' => $this->data['passengers']['adult'],
        'ChildQuantity' => $this->data['passengers']['child'],
        'InfantQuantity' => isset($this->data['passengers']['infant']) ? $this->data['passengers']['infant'] : 0,
        'VehicleQuantity' => $this->data['vehicle']['count'],
        'ChildAge' => [],
        'InfantAge' => [],
        'VehcileType' => [],
        'HavePets' => false
    ];

    // Add child age to params
    if ($this->data['passengers']['child'] !== 0) {

        for ($i = 0; $i < $this->data['passengers']['child']; $i++) {

            $childAge = $this->data['birthdates'][$i];
            $this->params['ChildAge'][] = ['Age' => $childAge];
        }
    }
```

Şekil 4: getRoutesRequest sınıfının setParams metodu

setParams metodunda yukarıda da görüldüğü üzere sefer tarihi, iskele IDleri ve yetişkin, çocuk ve bebek sayıları ile beraber yçocuk ve bebek için yaşları gerekli. Bende bu verileri kullanıcının site üzerinden girdiği değerleri göndererek bu isteğin doğru bir şekilde çalışmasını sağladım. Kullanıcıların girdiği değerler getRoutesRequest sınıfına gönderiliyor. Bu şekilde bu değerleri bende kullanabildim.

getRoutesResponse sınıfında ise göndermiş olduğum istekden gelen cevapları belli bir düzene göre bir dizinin içine kaydetme işlemini yaptım. Client sınıfı üzerinden gelen cevaba göre eğer hata varsa exception gönderecek şekilde, hata yoksa sonuçları listeyelecek şekilde response sınıfını yazdım.

```php
public function __construct($response, $data) {

    if(!isset($response->Result) || $response->Result === null){

        if(isset($response->Valid) && is_array($response->Valid)){

            if($response->Valid[0]->Value === null || $response->Valid[0]->Value === ''){
                $message = $response->Valid[0]->Key;
            }else{
                $message = $response->Valid[0]->Value;
            }
        }else{
            $message = '';
        }

        throw new GetRoutesErrorException($message);
    }

    $this->response = $response;
    $this->data = $data;
}
```

Şekil 5: getRoutesResponse constructor

Constructorda eğer gelen cevap boş veya null ise istekden gelen hata mesajını kaydedip GetRoutesErrorException metodunu çağırıyorum bu sayede kullanıcı hata mesajını görüp yeniden arama yapmaya ve başka bir arama yapmaya yönlendiriliyor.

```php
class GetRoutesErrorException extends BaseException implements ErturkExceptionInterface {

    public function __construct($message) {

        $level = 'error';

        $humanized_message = 'Sefer arama hatası: ' . $message;

        $provider_message = null;

        $extra = null;

        parent::__construct($level, $message, $humanized_message, $provider_message, $extra);
    }

}
```

Şekil 6: GetRoutesErrorException sınıfı

Sefer arama esnasında oluşan bir hata olduğunda kullanıcıya hata mesajı gösterilmesi burada yapıldı. Humanized_message değişkeni hatanın ekrana basılması için oluşturulan bir değişkendir.

```php
public function getResult() {

    // No routes found. Return empty result
    if ($this->response->Result === null) {
        return $this->result = [];
    }

    //Date formatters
    $dateFormatLong = new IntlDateFormatter('tr_TR', IntlDateFormatter::FULL, IntlDateFormatter::FULL);
    $dateFormatLong->setPattern('dd LLLL YYYY eeee');
    $dateFormatShort = new IntlDateFormatter('tr_TR', IntlDateFormatter::FULL, IntlDateFormatter::FULL);
    $dateFormatShort->setPattern('d LLLL eee');

    foreach ($this->response->Result as $i => $route) {

        // FerryName
        $shipName = $route->FerryName;

        // Create DateTime objects for this route's departure and arrival.
        $departure = new DateTime(substr($route->ServiceDate, 0, 10) . ' ' . $route->DepartureTime);
        $arrival = new DateTime(substr($route->ServiceDate, 0, 10) . ' ' . $route->ArrivalTime);

        // Create DateInterval object for route duration
        $duration = $route->TravelTime;

        $hours = floor($duration / 60);
        $minute = $duration - ($hours * 60);

        $duration_humanized = $hours . ' sa ' . $minute . ' dk';
```

Şekil 7: getRoutesResponse getResult metodu

getResult metodunda servisten gelen cevabın içindeki Result objesi foreach döngüsü ile içindeki her bir sefer için gelen bilgiler $result değişkeni içine daha sonra kullanılmak için kaydedildi.

```php
        $result = [
            'id' => $this->Generator->uuid(),
            'trip_number' => $route->ExpeditionID,
            'ferry' => [
                'type_code' => null,
                'type_name' => null,
                'name' => $shipName,
                'image' => 'http://www.erturk.com.tr/content/image/ferryboat/' . $route->FerryID . '/small/' . $route->Picture
            ],
            'firm' => [
                'id' => 3,
                'name' => 'Ertürk'
            ],
            'origin' => [
                'code' => $this->Converter->portIdToCode($this->data['originID']),
                'name' => $route->DeparturePort
            ],
            'destination' => [
                'code' => $this->Converter->portIdToCode($this->data['arrivalID']),
                'name' => $route->ArrivalPort
            ],
            'departure' => [
                'iso' => $departure->format('c'),
                'humanized' => [
                    'date' => $departure->format('d.m.Y'),
                    'time' => $departure->format('H:i'),
                    'short' => $dateFormatShort->format($departure),
                    'long' => $dateFormatLong->format($departure)
                ]
            ],
            'arrival' => [
                'iso' => $arrival->format('c'),
                'humanized' => [
                    'date' => $arrival->format('d.m.Y'),
                    'time' => $arrival->format('H:i'),
                    'short' => $dateFormatShort->format($arrival),
                    'long' => $dateFormatLong->format($arrival),
                ]
            ],
            'duration' => [
                'iso' => $duration,
                'time' => $duration,
                'humanized' => $duration_humanized
            ],
            'pricing' => null
        ];
```

Şekil 8: getRoutesResponse sınıfı getResult metodu devamı…

Her bir sefer getRoutesResposne sınıfında result değişkenine kaydedildi. Böylece kullanıcı ekranında gösterilmek üzere kaydedilmiş oldu. $result değişkininde sefer saati, feribot ismi, feribot tipi, fiyat gibi bilgiler servisten geliyor böylece bu bilgileri alıp kullanıcıya gösterebiliyoruz.

Temel olarak bu proje yaptığım işler bu şekildeydi. Her bir istek için ayrı ayrı request ve response sınıflarını oluşturdum. Exception sınıfları yine yaptım. Mesela sefer arama isteğinden başka, rezervasyon yapma isteği, ödeme istekleri gibi diğer durumlar için sınıflar yazdım.

Son olarak getRoutesRequest sınıfının nasıl çağrıldığını aşağıdaki görselde göstereceğim.

```
//Send search request for outgoing route
$OutgoingGetRoutesRequest = $this->RequestFactory->getRequest(GetRoutesRequest::class, $dataOutgoing);
$OutgoingGetRoutesResponse = $OutgoingGetRoutesRequest->send();
$outgoing_result = $OutgoingGetRoutesResponse->getResult();
```

Şekil 9: Sefer arama isteğinin çağrılması

Burada ilk satırda GetRoutesRequest sınıfı $dataOutgoing değişkeni gönderilerek oluşturulmaktadır. $dataOutgoin değişkeninde kullanıcının girmiş olduğu tarih, yolcu bilgisi gibi değerler bulunmaktadır. Daha sonra send metodu çağırılarak istek yapılmaktadır. En son olarak getRoutesResponse sınıfının getResult metosu çağırılarak $outgoing_result değişkenine kaydediliyor.

Bu şekilde ilk projemdeki görevlerimi tamamladım.

4.2 Laravel ve MVC Yapısını Kullanarak Web Sitesi Geliştirme

Bu projemizde entegre ettiğimiz feribot biletlerinin şirketin kendi bünyesinde satışları takip ettiği panele eklenmesi ve raporların yapılmasıydı. Bu projede iki temel iş yaptım. Birincisi satılan ve rezervasyonda kalan biletlerin listelenmesiydi. İkincisi ise günlük olarak satılan biletlerin raporlanması toplam bilet sayısı, toplam yolcu sayısı gibi verilerin oluşturulmasıydı.

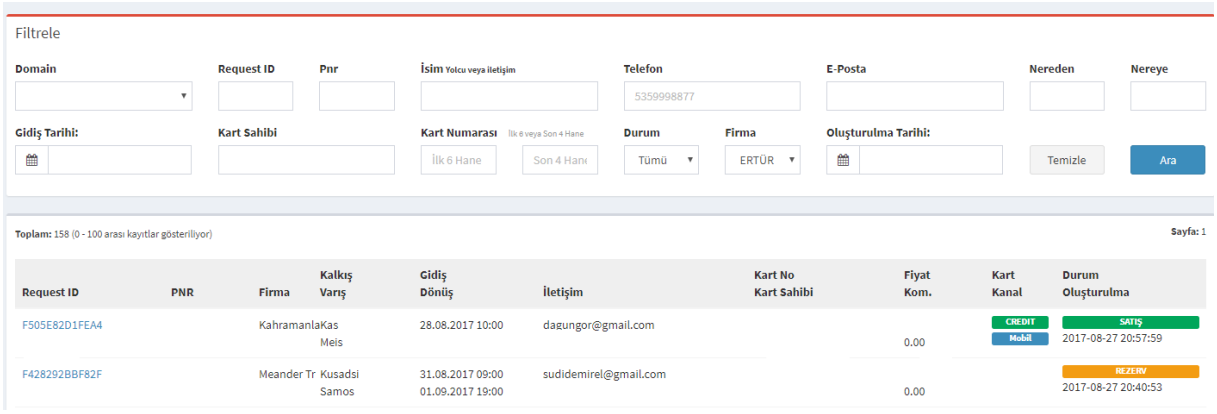
4.2.1 Kullandığım Teknolojiler

Bu projede kullandığım teknolojiler kısa açıklamaları ile şunlardır:

- **PHP**: Web siteleri ve web uygulamaları geliştirilmek için oluşturulmuş web tabanlı çalışan bir programlama dilidir.
- **HTML**: Web sayfaları oluşturmak ve düzenlemek için kullanılan bir işretleme dilidir.
- **Laravel**: Geliştirilmiş bir çok özellik ve yapıyı bünyesinde bulunduran PHP ve nesne tabanlı programlamadan yaralanan açık kaynak kodlu bir PHP frameworktur.
- **MVC**: Model-View-Controller yazılan kodların temel olarak daha düzenli ve anlaşılır olması için tasarlanan bir mimari yapıdır.

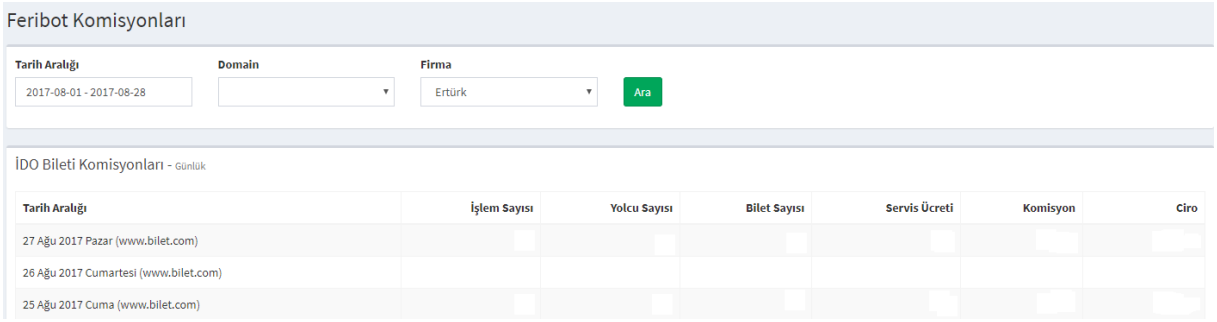
4.2.2 Arayüz Açıklamaları

İşlem yapılan tüm biletlerin listelendiği sayfa. Burada siteden yapılan feribor biletlerinin listelenmesi yapılıyor. Satılan ve rezervasyon yapılan biletler teker teker gösterilmektedir.



Şekil 10: Feribot biletlerinin listelenme ekranı

Yukarıda örnek olarak iki tane işlem gösterilmektedir. Yolcunun PNR, kalış – varış yerleri tarihi, fiyat, yolcu ismi gibi bilgiler gösterilmektedir. Bu bilgilerden bazılarını kaldırmak zorunda kaldım. Buradaki veriler vertabından sorgulanıp gösterilmektedir.



Şekil 11: Günlük satılan biletlerin raporlanması

Burada ise seçilen tarih aralığına göre gün gün satılan biletlerin sayısı ve toplam komisyon miktarları gösterilmektedir. Burdaki bilgileride kaldırmak zorunda kaldım.

4.2.3 Kodlar ve Açıklamaları

Bu projede raporlamaları yaparken MVC (Modal-View-Controller) yapısını kullandım. Verirabanı ile bağlantı kurmak için önce Modal tanımladım.

```php
class Booking extends Model{
    protected $connection = "mysql-bilet";
    protected $table = "ferry_bookings";

    protected static function boot() {
        parent::boot();

        static::addGlobalScope(new \Models\Scopes\DomainScope);
    }

    public function booking_items(){
        return $this->hasMany('Models\Ferry\BookingItem', 'request_id', 'request_id');
    }

    public function contact(){
        return $this->hasOne('Models\Ferry\Contact', 'request_id', 'request_id');
    }

    public function passengers(){
        return $this->hasMany('Models\Ferry\Passengers', 'request_id', 'request_id');
    }

    public function pos_transaction() {
        return $this->hasOne('Models\Pos\PosTransaction', 'order_id', 'request_id')->orderBy('id', 'DESC');
    }
}
```

Şekil 12: Booking Modal sınıfı

Bu sınıfta veritabanında kullanacağım tablo adını $table değişkenine kaydetme işlemini gerçekleştirdim.

```php
<div class="content">

    @include('tickets.ferry.partials.filter')

    <div class="row">
        <div class="col-xs-12">
            <div class="box">

                <div class="box-header">
                    <small><b>Toplam:</b> {{ $tickets->total() }}</small>
                    <small>({{ (($tickets->currentPage()-1)*100) }} - {{ ($tickets->currentPage()*100) }} arası kayıtlar gösteriliyor)</small>
                    <small class='pull-right'><b>Sayfa: </b> {{$tickets->currentPage()}}</small>
                </div>

                <style> table *{white-space: nowrap;}</style>

                <div class="box-body table-responsive">
                    <table class="table table-hover">
                        <thead style="background-color:#efefef;">
                        <th style="max-width:85px; overflow: hidden;">Request ID</th>
                        <!--th class="text-center">Channel</th-->
                        <!--th>Domain</th-->
                        <th>PNR</th>
                        <!--th>PNR</th-->
                        <th style="max-width: 35px;">Firma</th>
                        <th>Kalkış <br/> Varış</th>
                        <!--th>Varış</th-->
                        <th>Gidiş <br/> Dönüş</th>
                        <!--th>Dönüş</th-->
                        <!--th>İletişim</th-->
                        <th>İletişim</th>
                        <!--th class='text-right'>Telefon</th-->
                        <th>Kart No <br/> Kart Sahibi</th>
                        <!--th>Kart Sahibi</th-->

                        <th class='text-left'>Fiyat <br/> Kom.</th>
                        <!--th class='text-right'>Komisyon</th-->
                        <th class='text-left' style="max-width: 40px;">Kart <br/> Kanal</th>
                        <!--th class='text-center'>Pos</th-->
                        <th class='text-center'>Kanal</th-->
                        <th style="max-width: 85px;">Durum <br/> Oluşturulma</th>
                        <!--th>Oluşturulma</th-->
                        </thead>
```

Şekil 13: İşlem yapılan tüm biletlerin gösterildiği view.

View kısmında HTML kodları ile tablonun başlık kısmını oluşturdum. PNR, firma, gidiş – dönüş tarihi ve yerleri, iletişim gibi başlıklar oluşturuldu. Daha sonra tablonun body kısmı yapıldı ve tüm bilgiler ekrana basıldı.

```
<tbody>

    @foreach($tickets as $t)
    <tr>
        <td style="max-width: 85px; overflow: hidden;"><a href="{!! route('tickets.ferry.view', ['id' => $t->id]) !!}" target="_blank">{!! $t->request_id !!
        <!--td class='text-center'>
        @if($t->pos_transaction && $t->pos_transaction->channel == 'Web.Desktop')
        <div class='bg-orange center-block label'>Masaüstü</div>
        @elseif($t->pos_transaction && $t->pos_transaction->channel == 'Web.Mobile')
        <div class='bg-light-blue center-block label'>Mobil</div>
        @elseif($t->pos_transaction)
        <div class='bg-fuchsia center-block label'>App</div>
        @endif
        </td-->
        <!--td>{!! $t->domain !!}</td-->
        <td style="max-width: 50px;">{!! $t->pnr_number_outgoing !!} <br/> {!! $t->pnr_number_return !!}</td>
        <!--td>{!! $t->pnr_number_return !!}</td-->
        <td style="max-width: 25px;">{!! $t->outgoing_firm_name !!}</td>
        <td style="max-width: 50px; overflow: hidden;">{!! $t->outgoing_origin_name !!} <br/> {!! $t->outgoing_destination_name !!}</td>
        <!--td>{!! $t->outgoing_destination_name !!}</td-->
        <td style="max-width: 70px; overflow: hidden;">{!! date("d.m.Y H:i", strtotime($t->outgoing_departure)) !!} <br/> {!! $t->return_departure == null ?
        <!--td>{!! $t->return_departure == null ? "" : date("d.m.Y H:i", strtotime($t->return_departure)) !!}</td-->
        <!--td>{!! isset($t->contact) && $t->contact ? $t->contact->fullname : '' !!}</td-->
        <td style="max-width: 120px; overflow: hidden;">{!! isset($t->contact) && $t->contact ? $t->contact->email : '' !!} <br/> {!! isset($t->contact) &&
        <!--td class='text-right'>{!! isset($t->contact) && $t->contact ? $t->contact->phone : '' !!}</td-->
```

Şekil 14: İşlem yapılan tüm biletlerin gösterildiği view devamı…

Foreach ile veritabanından sorgulanan sonuçlar satır satır ekranda gösterilmek üzere tabloya eklendi. Böylece tüm bilgiler ekrana basılmış oldu.

```
// Ferry Tickets
Route::get('tickets/ferry', ['as' => 'tickets.ferry', 'uses' => 'Tickets\Ferry\TicketsController@index']);
Route::get('tickets/ferry/{id}/view', ['as' => 'tickets.ferry.view', 'uses' => 'Tickets\Ferry\TicketsController@view']);
```

Şekil 15: Laravel routes yapısı.

Laravel routes yapısında her bir url yapısı için controller içinde bir metod çağırılmaktadır. Burada da tickets/ferry url için TicketsController'undaki index metodu çağırılmıştır. Controllerde veritabanında sorgulama yapıp dönen cevabı viewin içine gönderilme işlemi yapılmaktadır.

```
class TicketsController extends Controller {

    private $ports = [
        'AMB' => 'Ambarlı',
        'ARM' => 'Armutlu',
        'ART' => 'Armutlu Tatil Köyü',
        'AVC' => 'Avcılar',
        'AVA' => 'Avşa Adası',
        'BDM' => 'Bandırma',
        'BSK' => 'Beşiktaş',
        'BST' => 'Bostancı',
        'BRS' => 'Bursa',
        'CNK' => 'Çınarcık',
        'ESY' => 'Esenköy',
        'KDK' => 'Kadıköy',
        'KAR' => 'Kartal',
        'KUM' => 'Kumla',
        'MAR' => 'Marmara Adası',
        'PEN' => 'Pendik',
        'TOP' => 'Topçular',
        'TZL' => 'Tuzla',
        'YLV' => 'Yalova',
        'YLD' => 'Yalova DO',
        'YKP' => 'Yenikapı'
    ];

    public function index(Request $request) {

        $input = $request->all();

        $this->data["tickets"] = Booking::with(['contact', 'passengers', 'pos_transaction'])->where(function ($q) use ($input) {

            if (isset($input["request_id"]) && $input["request_id"]) {
                $q->where('request_id', '=', $input["request_id"]);
            }

            if (isset($input["domain"]) && $input["domain"]) {
                $q->where('domain', '=', $input["domain"]);
            }
```

Şekil 16: Tickets Controller ve index metodu

```
if (isset($input["firm"]) && $input["firm"] != "0") {
    if ($input['firm'] == 'ido') {
        $q->where('provider', '=', 'ido');
    } elseif ($input['firm'] == 'budo') {
        $q->where('provider', '=', 'budo');
    } elseif ($input['firm'] == 'erturk') {
        $q->where('provider', '=', 'erturk');
    }
}
```

Şekil 17: TicketsController firma seçimi

Burada kullanıcı hangi firmanın biletlerini listelemek istiyorsa ona göre veritabanından sorgulama yapılabilmesi için gerekli düzenlemeler yapıldı. Eğer hiçbir firma seçilmezse bütün firmaların biletleri listelenmektedir.

5 SONUÇ

Bu yaz ikinci stajımı gerçekleştirme imkanı buldum. Çalışmış olduğum şirket yaklaşık henüz 2 yıllık bir şiket ve girişim seviyesinde bir firma olduğundan dolayı bu stajım boyunca daha kurumsal bir firma yerine girişim seviyesinde ve büyümekte olan bir şirkette işlerin nasıl yürüdüğüne dair izlenimler elde ettim. Bu gözlenimlerin özellikle ileride eğer bir girişim yapmak istersem neler ile karşılaşacağım hakkında önemli olacağını düşünüyorum.

Raporumda da belirttiğim üzere yazılım alanında kendimi geliştirme fırsatı elde ettim. Özellikle PHP ile web programlama ve mysql ile veritabanı konusunda yeni bilgiler edindim. Önceki stajımda çalıştığım şirket Oracle veritabanı kullanıyordu. Bu stajımda ise mysql kullandığım için bu iki konuda fikir sahibi oldum. Yine staj boyunca MVC yapısını kullanarak program geliştirme imkanı elde ettim. Bu sayede uzun süreden beri duyduğum bir konu hakkında geliştirme yaparak MVC mantığını anlamaya çalıştım. Ek olarak bir framework olan Laravel hakkında bilgi sahibi oldum. Frameworklerin mantığını ve geliştirme sırasına nasıl kolaylıklar sağladığını öğrendim. Son olarak şirket yazılımı geliştirirken bitbucket ve git kullanıyorlardı. Bende staj sürem boyunca bu teknolojiyi kullandığım için git yapısı ve özellikleri hakkında fikir ve tecrübe sahibi oldum.

Çalıştığım şirket eleman sayısı olarak çok fazla bir çalışanı olmadığında dolayı genelde her gün sabahları tek bir ekip olarak toplanıp o gün yapılması gerekenler ve önceki gün neler yaptıkları hakkında bir toplantı yapıyorlar. Bende tek stajyer olarak bu toplantıların içine dahil oldum ve bir kendimi bir firmanın çalışanı gibi hissedip sorumluluklarımın farkına varma imkanı buldum. Bu toplantıları önceki staj yaptığım firmadaki agile ve scrum metodolojisine benzetiyorum.

Bu stajımda iş dünyasında çalışma şartları ve ortamı hakkında fikirler edindim. Yeni insanlarla tanıştım ve bir takımın parçası oldum. Bilmediğim yeni teknolojiler öğrendim. Tenkokentlerde çalışma ortamlarını inceleme fırsatım oldu. Girişimler hakkında yeni bilgiler edindim.

Son olarak burada staj yapmayı diğer arkadaşlarıma tavsiye ederim. Çünkü, burada çalışanlar sizlere karşı çok sıcak ve yardımseverler. Bana her zaman yardımcı oldular. Takıldığımız yerlerde rahatça soru sorabiliyorduk. Ayrıca öğle aralarında çalışanlarla birlikte vakit geçirip sohbet etme fırsatımız oluyordu. E-ticaret sektörü hakkında da genel birkaç bilgi öğrenebilirsiniz. Özellikle bir girişimin içinde bulunmak isteyen arkadaşlara burayı tavsiye edbilirim.

6 REFERANSLAR

- Digital Trade A.Ş. web sayfası: www.digitaltrade.com.tr
- PHP web sayfası: http://php.net/manual/tr/index.php
- Laravel 5.2 dokümantasyon: https://laravel.com/docs/5.2
- Netbeans web sayfası: https://netbeans.org/
- JSON tutorial: https://www.w3schools.com/js/js_json_intro.asp
- MVC tutorial: http://requiremind.com/a-most-simple-php-mvc-beginners-tutorial/
- Guzzle dokümantason: http://docs.guzzlephp.org/en/stable/

7 ŞEKİLLER TABLOSU

8 SUMMARY

I had the opportunity to make this internship 20 working days in the Digital Trade company located at Istanbul University TechnoPark Building. Digital Trade is a company established in 2015 in e-commerce sector. Digital Trade have Bilet.com and Elbise.com brands. The company develops its own software. It consists of IT, Marketing and Customer service departments. I made my internship at IT department.

There is a daily meeting within the company and everyone tells what they did yesterday and what they will do on that day. These meetings are of great importance for follow works. I attended this meeting throughout my internship, so I got the opportunity to be involved in a group.

During my internship I worked on two projects. The first was the integration of the tickets of a ferry company where the company had an agreement. The project was to integrate the information from the ferry company via a web service to the site of Bilet.com. I have provided the rewritten request and response classes for the new webservice, which were previously written throughout to another project. I also rearranged the exception classes.

When I was doing these tasks, there were people I could ask questions and helped me all the time. The web service was working in JSON format. The requests and response I sent were always in JSON format.

The second project was to report all Greek island tickets made on Bilet.com after the completion of the integration of the ferry tickets to the Greek islands. Tickets that were sold and booked were questioned and reported from the database. I also took part in reporting the number of tickets sold daily, the number of passengers and the total commissions. So the company will see how many tickets you sell on a daily basis and how many transactions are todos.

So I finished my internship this summer. Throughout my internship, I had the opportunity to observe the workings in the technoparks. I had an idea how private companies cooperate. Finally, I worked in a project and developed myself in the field of software.