АНОТАЦІЯ

Випускна кваліфікаційна робота на тему «Розробка web-додатку управління логістичними послугами» містить 43 сторінки, 16 рисунків та 1 додаток. Список використаних джерел включає 9 пунктів.

Об'єктом дослідження є діяльність логістичних компаній.

Предметом дослідження є методи та алгоритми розробки web-додатку.

Метою дослідження випускної кваліфікаційної роботи є дослідження можливостей у сфері web-розробки, набуття навичок щодо застосування мов web-програмування, середовищ розробки та спеціалізованих текстових редакторів, систем управління базами даних тощо.

Завдання дослідження: Вибір мови програмування, аналіз існуючих підходів до розробки візуального web-додатку управління логістичними послугами.

Методи дослідження: Аналіз технологій для створення web-додатку.

Перший розділ мітить основні поняття для розуміння потрібності продажу логістичних послуг через веб-сайт.

Другий розділ розробка web-додатку з архітектурою «клієнт-сервер».

Третій розділ безпосереднє створення самого сайту з використанням фреймворк Symfony, створення моделі бази данних.

ANNOTATION

The final qualifying work on the topic "Development of a web-application for logistics services management" contains 43 pages, 16 figures and 1 appendix. The list of used sources includes 9 items.

The object of research is the activities of logistics companies.

The subject of research is the methods and algorithms of web-application development.

The purpose of the final qualification work is to study the possibilities in the field of web-development, acquisition of skills in the use of web-programming languages, development environments and specialized text editors, database management systems, etc.

Research objectives: Choice of programming language, analysis of existing approaches to the development of a visual web-application for logistics services management.

Research methods: Analysis of technologies for creating a web-application.

The first section outlines the basic concepts for understanding the need to sell logistics services through a website.

The second section is the development of a web-application with a client-server architecture.

The third section is the direct creation of the site itself using the Symfony framework, creating a database model.

ДОДАТКИ

```
<?php

namespace App\Repository;

use App\Entity\Car;
use Doctrine\Bundle\DoctrineBundle\Repository\ServiceEntityRepository;
use Doctrine\Persistence\ManagerRegistry;

/**
 * @method Car|null find($id, $lockMode = null, $lockVersion = null)
 * @method Car|null findOneBy(array $criteria, array $orderBy = null)
 * @method Car[]    findAll()
 * @method Car[]    findBy(array $criteria, array $orderBy = null, $limit = null, $offset = null)
 */
class CarRepository extends ServiceEntityRepository
{
    public function __construct(ManagerRegistry $registry)
    {
        parent::__construct($registry, Car::class);
    }

    public function getListWithPagination(int $limit = 1, int $offset = 0): array
    {
        return $this->createQueryBuilder('c')
            ->setMaxResults($limit)
            ->setFirstResult($offset)
```

```
            ->getQuery()
            ->getResult()
            ;
    }
}


<?php


namespace App\Repository;


use App\Entity\Country;
use Doctrine\Bundle\DoctrineBundle\Repository\ServiceEntityRepository;
use Doctrine\Persistence\ManagerRegistry;


/**
 * @method Country|null find($id, $lockMode = null, $lockVersion = null)
 * @method Country|null findOneBy(array $criteria, array $orderBy = null)
 * @method Country[]    findAll()
 * @method Country[]    findBy(array $criteria, array $orderBy = null, $limit = null, $offset = null)
 */
class CountryRepository extends ServiceEntityRepository
{
    public function __construct(ManagerRegistry $registry)
    {
        parent::__construct($registry, Country::class);
    }
}
```

```php
<?php

namespace App\Form;

use App\Entity\Car;
use Symfony\Component\Form\AbstractType;
use Symfony\Component\Form\FormBuilderInterface;
use Symfony\Component\OptionsResolver\OptionsResolver;
use Symfony\Component\Form\Extension\Core\Type\FileType;
use Symfony\Component\Validator\Constraints\File;
use Vich\UploaderBundle\Form\Type\VichImageType;

class CarType extends AbstractType
{
    public function buildForm(FormBuilderInterface $builder, array $options)
    {
        $builder
            ->add('imageFile', VichImageType::class, [
                'label' => 'image',

                // unmapped means that this field is not associated to any entity property
                'mapped' => true,

                // make it optional so you don't have to re-upload the PDF file
                // every time you edit the Product details
```

```
            'required' => false,


            // unmapped fields can't define their validation using annotations
            // in the associated entity, so you can use the PHP constraint classes
            'constraints' => [
                new File([
                    'maxSize' => '1024k',
                    'mimeTypes' => [
                        'image/jpeg',
                        'image/png',
                    ],
                    'mimeTypesMessage' => 'Please upload a valid PDF document',
                ])
            ],
        ])
        ->add('make')
        ->add('model')
        ->add('engine')
        ->add('fuel')
        ->add('transmission')
        ->add('year')
    ;
}


public function configureOptions(OptionsResolver $resolver)
{
    $resolver->setDefaults([
        'data_class' => Car::class,
    ]);
```

```
    }
}
```

```
<?php

namespace App\Entity;

use App\Repository\CountryRepository;
use Doctrine\ORM\Mapping as ORM;

/**
 * @ORM\Entity(repositoryClass=CountryRepository::class)
 */
class Country
{
    /**
     * @ORM\Id()
     * @ORM\GeneratedValue()
     * @ORM\Column(type="integer")
     */
    private $id;

    /**
     * @ORM\Column(type="string", length=255, nullable=true)
     */
    private $name;
```

```
    /**
     * @ORM\Column(type="float", nullable=true)
     */
    private $shipping_price;

    public function getId(): ?int
    {
        return $this->id;
    }

    public function getName(): ?string
    {
        return $this->name;
    }

    public function setName(?string $name): self
    {
        $this->name = $name;

        return $this;
    }

    public function getShippingPrice(): ?float
    {
        return $this->shipping_price;
```

```
    }

    public function setShippingPrice(?float $shipping_price): self
    {
        $this->shipping_price = $shipping_price;

        return $this;
    }
}
```

```
<?php

namespace App\Entity;

use App\Repository\CarRepository;
use Doctrine\ORM\Mapping as ORM;
use Symfony\Component\HttpFoundation\File\File;
use Symfony\Component\HttpFoundation\File\UploadedFile;
use Vich\UploaderBundle\Entity\File as EmbeddedFile;
use Vich\UploaderBundle\Mapping\Annotation as Vich;

/**
 * @ORM\Entity(repositoryClass=CarRepository::class)
 * @Vich\Uploadable
 */
class Car
```

```
{
    /**
     * @ORM\Id()
     * @ORM\GeneratedValue()
     * @ORM\Column(type="integer")
     */
    private $id;


    /**
     * @ORM\Column(type="string", length=255, nullable=true)
     */
    private $make;


    /**
     * @ORM\Column(type="string", length=255, nullable=true)
     */
    private $model;


    /**
     * @ORM\Column(type="float", nullable=true)
     */
    private $engine;


    /**
     * @ORM\Column(type="string", length=255, nullable=true)
     */
    private $fuel;
```

```
    /**
     * @ORM\Column(type="string", length=255, nullable=true)
     */
    private $transmission;


    /**
     * NOTE: This is not a mapped field of entity metadata, just a simple property.
     *
     * @Vich\UploadableField(mapping="car_image", fileNameProperty="image.name", size="image.size", mimeType="image.mimeType", originalName="image.originalName", dimensions="image.dimensions")
     *
     * @var File|null
     */
    private $imageFile;


    /**
     * @ORM\Embedded(class="Vich\UploaderBundle\Entity\File")
     *
     * @var EmbeddedFile
     */
    private $image;


    /**
     * @ORM\Column(type="integer", nullable=true)
     */
    private $year;
```

```
    /**
     * @ORM\Column(type="float", nullable=true)
     */
    private $price;


    public function __construct()
    {
        $this->image = new EmbeddedFile();
    }


    /**
     * If manually uploading a file (i.e. not using Symfony Form) ensure an instance
     * of 'UploadedFile' is injected into this setter to trigger the  update. If this
     * bundle's configuration parameter 'inject_on_load' is set to 'true' this setter
     * must be able to accept an instance of 'File' as the bundle will inject one here
     * during Doctrine hydration.
     *
     * @param File|UploadedFile|null $imageFile
     */
    public function setImageFile(?File $imageFile = null)
    {
        $this->imageFile = $imageFile;
//
//        if (null !== $imageFile) {
//            // It is required that at least one field changes if you are using doctrine
//            // otherwise the event listeners won't be called and the file is lost
//            $this->updatedAt = new \DateTimeImmutable();
```

```
//      }
    }

    public function setImage(EmbeddedFile $image): void
    {
        $this->image = $image;
    }

    public function getImage(): ?EmbeddedFile
    {
        return $this->image;
    }

    public function getId(): ?int
    {
        return $this->id;
    }

    public function getMake(): ?string
    {
        return $this->make;
    }

    public function setMake(?string $make): self
    {
        $this->make = $make;
```

```
    return $this;
}

public function getModel(): ?string
{
    return $this->model;
}

public function setModel(string $model): self
{
    $this->model = $model;

    return $this;
}

public function getEngine(): ?float
{
    return $this->engine;
}

public function setEngine(?float $engine): self
{
    $this->engine = $engine;
```

```
        return $this;
    }

    public function getFuel(): ?string
    {
        return $this->fuel;
    }

    public function setFuel(?string $fuel): self
    {
        $this->fuel = $fuel;

        return $this;
    }

    public function getTransmission(): ?string
    {
        return $this->transmission;
    }

    public function setTransmission(?string $transmission): self
    {
        $this->transmission = $transmission;

        return $this;
    }
```

```
public function getYear(): ?int
{
    return $this->year;
}

public function setYear(?int $year): self
{
    $this->year = $year;

    return $this;
}

/**
 * @return File|null
 */
public function getImageFile(): ?File
{
    return $this->imageFile;
}

public function getPrice(): ?float
{
    return $this->price;
}
```

```
    public function setPrice(?float $price): self
    {
        $this->price = $price;

        return $this;
    }
}

<?php

namespace App\DataFixtures;

use App\Entity\Car;
use App\Entity\Country;
use Doctrine\Bundle\FixturesBundle\Fixture;
use Doctrine\Common\Persistence\ObjectManager;
use Symfony\Component\HttpFoundation\File\UploadedFile;

class CountryDataFixtures extends Fixture
{
    private const DATA_SET =
        [
            'USA' => 3000,
            'Canada' => 2700,
            'Poland' => 300,
            'Germany' => 500,
            'Japan' => 3700,
```

```
        'Korea' => 4000,
        'France' => 550,
        'Italy' => 550,
    ];


  public function load(ObjectManager $manager)
  {
      foreach (self::DATA_SET as $key => $value) {
          $manager->persist(
              (new Country())->setName($key)->setShippingPrice($value)
          );
      }


      $manager->flush();
  }
}


<?php


namespace App\DataFixtures;


use App\Entity\Car;
use Doctrine\Bundle\FixturesBundle\Fixture;
use Doctrine\Common\Persistence\ObjectManager;
use Symfony\Component\HttpFoundation\File\UploadedFile;
use function sprintf;
```

```
class CarDataFixtures extends Fixture
{
    private const DATA_SET =
        [
            0 => [
                'make' => 'Toyota',
                'model' => 'Camry',
                'engine' => 3.5,
                'fuel' => 'бензин',
                'transmission' => 'автоматическая',
                'year' => 2019,
                'price' => 24000,
                'image' => 'camry19.jpg',
            ],
            1 => [
                'make' => 'Toyota',
                'model' => 'Corolla',
                'engine' => 2.0,
                'fuel' => 'бензин',
                'transmission' => 'автоматическая',
                'year' => 2013,
                'price' => 16000,
                'image' => 'corolla13.jpg',
            ],
            2 => [
                'make' => 'Toyota',
                'model' => 'Corolla',
                'engine' => 2.5,
```

```
    'fuel' => 'бензин',
    'transmission' => 'автоматическая',
    'year' => 2016,
    'price' => 22000,
    'image' => 'corolla16.jpg',
  ],
  3 => [
    'make' => 'Ford',
    'model' => 'Focus',
    'engine' => 1.0,
    'fuel' => 'бензин',
    'transmission' => 'автоматическая',
    'year' => 2018,
    'price' => 14000,
    'image' => 'focus18.jpg',
  ],
  4 => [
    'make' => 'Ford',
    'model' => 'Focus',
    'engine' => 2.5,
    'fuel' => 'бензин',
    'transmission' => 'автоматическая',
    'year' => 2019,
    'price' => 21000,
    'image' => 'focus19.jpg',
  ],
  5 => [
    'make' => 'Ford',
    'model' => 'Fusion',
    'engine' => 2.0,
```

```
    'fuel' => 'бензин',
    'transmission' => 'автоматическая',
    'year' => 2017,
    'price' => 23000,
    'image' => 'fusion17.jpg',
  ],
  6 => [
    'make' => 'Volkswagen',
    'model' => 'Golf VII',
    'engine' => 2.0,
    'fuel' => 'бензин',
    'transmission' => 'автоматическая',
    'year' => 2018,
    'price' => 15000,
    'image' => 'golfVII18.jpg',
  ],
  7 => [
    'make' => 'Hyundai',
    'model' => 'Elantra',
    'engine' => 2.0,
    'fuel' => 'бензин',
    'transmission' => 'автоматическая',
    'year' => 2018,
    'price' => 19000,
    'image' => 'hyundai-elantra18.jpg',
  ],
  8 => [
    'make' => 'Mercedes-Benz',
    'model' => 'cls',
    'engine' => 2.5,
```

```
    'fuel' => 'бензин',
    'transmission' => 'автоматическая',
    'year' => 2017,
    'price' => 30000,
    'image' => 'mersedes-cls17.jpg',
  ],
  9 => [
    'make' => 'Mercedes-Benz',
    'model' => 'cls',
    'engine' => 3.5,
    'fuel' => 'бензин',
    'transmission' => 'автоматическая',
    'year' => 2018,
    'price' => 38000,
    'image' => 'mersedes-cls18.jpg',
  ],
  10 => [
    'make' => 'Mercedes-Benz',
    'model' => 'cls',
    'engine' => 3.0,
    'fuel' => 'бензин',
    'transmission' => 'автоматическая',
    'year' => 2020,
    'price' => 60000,
    'image' => 'mersedes-cls20.jpg',
  ],
  11 => [
    'make' => 'Mercedes-Benz',
    'model' => 'gle',
    'engine' => 3.0,
```

```
            'fuel' => 'бензин',
            'transmission' => 'автоматическая',
            'year' => 2020,
            'price' => 80000,
            'image' => 'mersedes-gle20.jpg',
        ],
        12 => [
            'make' => 'Mercedes-Benz',
            'model' => 'gle-coupe',
            'engine' => 5.0,
            'fuel' => 'бензин',
            'transmission' => 'автоматическая',
            'year' => 2019,
            'price' => 92000,
            'image' => 'mersedes-gle-coupe19.jpg',
        ],
    ];


public function load(ObjectManager $manager)
{
    foreach (self::DATA_SET as $item) {
        $car = (new Car())
            ->setMake($item['make'])
            ->setModel($item['model'])
            ->setEngine($item['engine'])
            ->setFuel($item['fuel'])
            ->setTransmission($item['transmission'])
            ->setYear($item['year'])
            ->setPrice($item['price'])
```

```
            ;


            $car->setImageFile(
                new UploadedFile(
                    sprintf('%s/../../public/files/images/%s',__DIR__, $item['image']),
                    $item['image'],
                    null,
                    null,
                    true
                )
            );
            $manager->persist(
                $car
            );
        }


        $manager->flush();
    }
}


<?php


namespace App\Controller;


use App\Repository\CountryRepository;
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Response;
```

```
use Symfony\Component\Routing\Annotation\Route;


class DeliveryCostController extends AbstractController
{
    /**
     * @Route("/delivery", name="delivery-cost")
     * @return Response
     */
    public function index(CountryRepository $countryRepository): Response
    {
        $countries = $countryRepository->findAll();


        return $this->render('delivery_cost.html.twig', ['countries'=> $countries]);
    }
}
<?php


namespace App\Controller;


use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;


class DefaultController extends AbstractController
{
    /**
     * @Route("/", name="index")
```

```
     * @return Response
     */
    public function index(): Response
    {
        return $this->render('index.html.twig');
    }
}
<?php


namespace App\Controller;


use App\Entity\Car;
use App\Form\CarType;
use App\Repository\CarRepository;
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;


/**
 * @Route("/car")
 */
class CarController extends AbstractController
{
    private const LIMIT_PER_PAGE = 6;


    /**
```

```
     * @Route("/", name="car_index", methods={"GET"})
     */
    public function index(CarRepository $carRepository, Request $request): Response
    {
        $offset = $request->query->getInt('offset', 0);
        $cars = $carRepository->getListWithPagination(self::LIMIT_PER_PAGE, $offset);
        $total = $carRepository->count([]);


        return $this->render(
            'car/index.html.twig',
            [
                'cars' => $cars,
                'total' => $total,
                'offset' => $offset,
                'limit' => self::LIMIT_PER_PAGE,
            ]
        );
    }


    /**
     * @Route("/new", name="car_new", methods={"GET","POST"})
     */
    public function new(Request $request): Response
    {
        $car = new Car();
        $form = $this->createForm(CarType::class, $car);
        $form->handleRequest($request);
```

```
        if ($form->isSubmitted() && $form->isValid()) {
            $entityManager = $this->getDoctrine()->getManager();
            $entityManager->persist($car);
            $entityManager->flush();

            return $this->redirectToRoute('car_index');
        }

        return $this->render(
            'car/new.html.twig',
            [
                'car' => $car,
                'form' => $form->createView(),
            ]
        );
    }

    /**
     * @Route("/{id}", name="car_show", methods={"GET"})
     */
    public function show(Car $car): Response
    {
        return $this->render(
            'car/show.html.twig',
            [
                'car' => $car,
            ]
        );
```

```
}

/**
 * @Route("/{id}/edit", name="car_edit", methods={"GET","POST"})
 */
public function edit(Request $request, Car $car): Response
{
    $form = $this->createForm(CarType::class, $car);
    $form->handleRequest($request);

    if ($form->isSubmitted() && $form->isValid()) {
        $this->getDoctrine()->getManager()->flush();

        return $this->redirectToRoute('car_index');
    }

    return $this->render(
        'car/edit.html.twig',
        [
            'car' => $car,
            'form' => $form->createView(),
        ]
    );
}

/**
 * @Route("/{id}", name="car_delete", methods={"DELETE"})
```

```
     */
    public function delete(Request $request, Car $car): Response
    {
        if ($this->isCsrfTokenValid('delete' . $car->getId(), $request->request->get('_token'))) {
            $entityManager = $this->getDoctrine()->getManager();
            $entityManager->remove($car);
            $entityManager->flush();
        }


        return $this->redirectToRoute('car_index');
    }
}
```

```
{% extends 'base.html.twig' %}


{% block body %}


    <div class="container">


        <div class="row" style="height: 30vh;">
            <p style="font-size: 13pt; text-align: center;  padding-top: 50px ">
                Здесь вы можете поссчитать итоговую стоимость автомобиля. </br>Внимание! Это приблизительная стоимость, за точной суммой обратитесь к менеджеру.
                Для просчета условной итоговой стоимости заполните давнные.
            </p>
        </div>
        <div style=" background: url('/files/images/logotypes/plane-ship-car.jpg') no-repeat center/cover; min-height: 500px; padding: 100px 0 100px 0">
```

```
<div class="row" style="text-align: center; margin: 30px 0 0 0 ">
  <div class="col-6">Объем двигателя в куб. см.</div>
  <div class="col-6">
    <div style="width: 150px">
      <input type="text" class="form-control" id="engine" aria-describedby="emailHelp" placeholder="куб.см.">
    </div>
  </div>
</div>
<div class="row" style="text-align: center; margin: 30px 0 0 0 ">
  <div class="col-6">Кол-во лет автомобилю</div>
  <div class="col-6">
    <div style="width: 150px">
      <input type="text" class="form-control" id="years" aria-describedby="emailHelp" placeholder="кол-во лет">
    </div>
  </div>
</div>
<div class="row" style="text-align: center; margin: 30px 0 0 0 ">
  <div class="col-6">Вид топлива</div>
  <div class="col-6">
    <div style="width: 150px">
      <select class="form-control" id="fuel">
        <option value="petrol" selected>Бензин</option>
        <option value="diesel">Дизель</option>
      </select>
    </div>
  </div>
</div>
<div class="row" style="text-align: center; margin: 30px 0 0 0 ">
  <div class="col-6">Выберите страну</div>
```

```
            <div class="col-6">
                <div style="width: 150px">
                    <select class="form-control" id="country">
                        {% for country in countries %}
                            <option value='{{ country.shippingPrice }}'>{{ country.name }}</option>
                        {% endfor %}
                    </select>


                </div>
            </div>
        </div>
        <div class="row" style="text-align: center; margin: 30px 0 0 0 ">
            <div class="col-6">Укажите начальную стоимость авто.</div>
            <div class="col-6">
                <div style="width: 150px">
                    <input type="text" class="form-control" id="startPrice" aria-describedby="emailHelp"
placeholder="$">
                </div>
            </div>
        </div>
        <div class="row" style="text-align: center; margin: 30px 0 0 0 ">
            <div class="col-3"></div>
            <div class="col-3">
                <div style="width: 150px">
                    <button type="button" id="calculate" class="btn btn-dark">Посчитать</button>
                </div>
            </div>
            <div class="col-6"></div>
        </div>
    </div>
```

```
    <div style="height: 50px" id="responseAmount"></div>
  </div>
{% endblock %}
{% block javascripts %}


  <script type="text/javascript">


    $(function () {
      $('#calculate').on('click', function (e) {
        let engineSize = parseFloat($('#engine').val());
        if (!engineSize || isNaN(engineSize)) {
          alert("Укажите правильный объем двигателя");
          return;
        }
        let yearsCount = parseInt($('#years').val());
        if (!yearsCount || isNaN(engineSize)) {
          alert("Укажите сколько лет автомобилю");
          return;
        } else if (yearsCount === 0) {
          yearsCount = 1
        }
        let fuel = $("#fuel option:selected").val();
        if (!fuel) {
          alert("Укажите вид топлива");
          return;
        }
        let countryShippingPrice = parseInt($('#country').val());
        if (!countryShippingPrice || isNaN(countryShippingPrice)) {
          alert("Укажите страну");
```

```
            return;
        }
        let startPrice = parseFloat($('#startPrice').val());
        if (!startPrice || isNaN(startPrice)) {
            alert("Укажите правильный объем двигателя");
            return;
        }
        const amountWithoutCountryShipping = parseFloat(startPrice) +
parseFloat(getTaxAmount(engineSize, yearsCount, fuel, startPrice));
        const totalAmount = parseFloat(countryShippingPrice) +
parseFloat(amountWithoutCountryShipping) + parseFloat(1000);


        const messageResultAmount =
            'Учитывая НДС, пошлинную и таможенные сборы Ваш автомобиль будет стоить '
            + amountWithoutCountryShipping
            + '. С доставкой и услугами компании '
            + totalAmount
        ;


        $('#responseAmount').text(messageResultAmount);
    });


});


function getTaxAmount(engineSize, countYears, fuelType, startPrice) {
    const ndsPercent = 20;
    const tollPercent = 10;          //poshlinaya
    const retirementFeePercent = 5;    //pensionniy
```

```
        const axcisTax = getAxcisTaxAmount(engineSize, countYears, fuelType, startPrice,);
        const tollTax = startPrice / 100 * tollPercent;
        const ndsTax = (startPrice + axcisTax + tollTax) / 100 * ndsPercent;
        const retirementFeeTax = startPrice / 100 * retirementFeePercent;


        return parseFloat(axcisTax) + parseFloat(tollTax) + parseFloat(ndsTax) +
parseFloat(retirementFeeTax);
    }


    function getAxcisTaxAmount(engineSize, countYears, fuelType) {
        let engineTax = 0;
        if (fuelType == 'petrol') {
            engineTax = engineSize < 3000 ? 50 : 100;
        } else if (fuelType == 'diesel') {
            engineTax = engineSize < 3000 ? 75 : 150;
        }
        const engineCoefficient = engineSize / 1000;
        const returnCoefficient = countYears;


        return engineTax * engineCoefficient * returnCoefficient;
    }
  </script>
{% endblock %}
```

Київський національний торговельно-економічний університет

Кафедра інженерії програмного забезпечення та кібербезпеки

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Розробка сайту з продажу логістичних послуг»

Студента 4 курсу, 6 групи, спеціальності 121 «Інженерія програмного забезпечення» _____ підпис студента Євдокименка Володимира Андрійовича

Науковий керівник кандидат технічних наук, доцент _____ підпис керівника Харченко Олександр Анатолійович

Гарант освітньої програми кандидат технічних наук, доцент _____ підпис керівника Цензура Микола Олександрович

КИЇВ – 2020

Зміст

ВСТУП

Логістика і транспортування – одна з найбільши затребуваних на ринку послуг, що надаються. У свою чергу, створення власного web-додатку найпростиший спосіб систематизувати і зробити більш зручним процеси взаємодій з клієнатми.

Як правило, в компаніях, діяльність яких безпосередньо повязана з транспортними послугами, не надто зациклені на ретельного опрацювання функцій сайту і зупиняють свій вибір переважно на сайтах-візитках, зозміщуючи там лише свої контакти. При цьому, як правило, вибираються вже готові платформи з шаблонами. Незважаючи на очевидність подібного підходу, сайт виходить дуже схожим на сотні інших. З цієї причини компанія втрачає шанси стати впізнаваною пряме перешкода до подальшего розширення бізнесу.

Об'єктом дослідження є діяльність логістичних компаній.

Предметом дослідження є методи та алгоритми розробки web-додатку.

Метою дослідження випускної кваліфікаційної роботи є дослідження можливостей у сфері web-розробки, набуття навичок щодо застосування мов web-програмування, середовищ розробки та спеціалізованих текстових редакторів, систем управління базами даних тощо.

Завдання дослідження: Вибір мови програмування, аналіз існуючих підходів до розробки візуального web-додатку управління логістичними послугами.

Методи дослідження: Аналіз технологій для створення web-додатку.

					КНТЕУ 121 06-05.БР			
Зм.	Аркуш	№ докум	Підпис	Дата				
Зав. кафедри		Криворучко О.В.			Розробка web-додатку управління логістичними послугами	Стадія	Аркуш	Аркушів
Керівник		Харченко О.А.				В	4	43
Гарант		Цензура М. О.				Факультет інформаційних технологій, 4 курс, 6 група		
Розроб.		Євдокименко В.А			Вступ			

РОЗДІЛ 1.
З ЧОГО СКЛАДАЄТЬСЯ САЙТ, КОМУ ВІН ПОТРІБЕН ТА ЯК ЙОГО СТВОРИТИ

1.1 Сайт

Сайт – Сайт, або веб-сайт (від англ. Website: web – «павутина, мережа» і site – «місце», буквально «місце, сегмент, частина в мережі»), – одна або кілька логічно пов'язаних між собою веб-сторінок; також місце розташування контенту сервера. Зазвичай сайт в Інтернеті являє собою масив пов'язаних даних, що має унікальну адресу і сприймається користувачами як єдине ціле.

Веб-сайти називаються так, тому що доступ до них відбувається по протоколу HTTP.

Веб-сайт, як система електронних документів (файлів даних і коду) може належати приватній особі або організації і бути доступним в комп'ютерній мережі під загальним доменним ім'ям і IP-адресою або локально на одному комп'ютері. У статті журналу «Господарство і право» також було висловлено думку, що кожен сайт має свою назву, яке при цьому не слід плутати з доменним ім'ям. З точки зору авторського права сайт є складовим твором, відповідно назва сайту підлягає охороні поряд з назвами всіх інших творів. Всі сайти в сукупності складають Всесвітню павутину, де комунікація об'єднує сегменти інформації світової спільноти в єдине ціле - базу даних і комунікації планетарного масштабу. Для прямого доступу клієнтів до сайтів на серверах був спеціально розроблений протокол HTTP.

Люди і компанії створюють сайти для різних цілей: щоб продавати товари та послуги, розміщувати і знаходити інформацію, отримувати знання, спілкуватися з іншими користувачами, розважатися. Кожен сам знаходить причину створити сайт, перерахувати їх всі не вийде. Компанії сайт стане в

нагоді, щоб створити репутацію, знайти нових клієнтів, збільшити продажі. На корпоративному сайті можна розповісти про діяльність компанії і її співробітників, представити асортимент товарів або послуг, провести рекламну кампанію або відповісти на запитання користувачів. Приватній особі сайт стане в нагоді, щоб знайти роботу або залучити нових клієнтів. На особистому сайті можна поділитипрофесійним досвідом, прорекламувати себе або свій бізнес, продавати товари або послуги в онлайні. Сторінки сайтів – це набір текстових файлів, розмічених мовою HTML. Ці файли, будучи завантаженими відвідувачем на його комп'ютер, розуміються і обробляються браузером і виводяться на засіб відображення користувача (монітор, екран КПК, принтер або синтезатор мови). Мова HTML дозволяє форматувати текст, розрізняти в ньому функціональні елементи, створювати гіпертекстові посилання (гіперпосилання) і вставляти в сторінку зображення, звукозаписи і інші мультимедійні елементи. Відображення сторінки можна змінити додаванням стилів на мові CSS, що дозволяє централізувати в певному файлі всі елементи форматування (розмір і колір заголовних букв 2-го рівня, розмір і вид блоку вставки і інше) або сценаріїв на мові JavaScript, за допомогою якого є можливість переглядати сторінки з подіями або діями. Сторінки сайтів можуть бути простим статичним набором файлів або створюватися спеціальною комп'ютерною програмою на сервері. Вони можуть бути або зроблені на замовлення для окремого сайту, або бути готовим продуктом, розрахованим на певний клас сайтів. Деякі з них можуть забезпечити власнику сайту можливість гнучкої настройки структуризації і виведення інформації на веб-сайті. Такі керуючі програми називаються системами управління змістом (CMS). Сайти можуть містити підрозділи, орієнтовані цілком на ту чи іншу аудиторію. У цьому випадку такі розділи називають версіями сайту. Аудиторія може відрізнятися по виду використовуваного

обладнання, по використовуваному мови аудиторії. Наприклад, відомі так звані мобільні версії сайту, призначені для роботи з ними з використанням смартфона. Сайти можуть мати мовні версії (російськомовна, англомовна і інші). Певний клас сайтів інакше називають інтернет-представництвом людини чи організації. Як коментар до заслання може бути сторінка-візитка на повнофункціональному сайті (порталі). Коли говорять «своя сторінка в Інтернеті», то мають на увазі цілий сайт або особисту сторінку в складі чужого сайту (портал). Крім сайтів (порталів), в мережі Інтернет також доступні WAP-сайти для мобільних телефонів. Спочатку сайти представляли собою сукупність статичних документів, наприклад – сайт-візитка. У міру розвитку комунікацій, кількість внутрішніх і зовнішніх посилань збільшувалася. Сайт став виконувати не тільки роль довідки, анотації, а й функціонального офісу, новинного або медійного центру. В даний час більшості з них властива динамічність і інтерактивність. Для таких випадків фахівці використовують термін веб-додаток – готовий програмний комплекс для вирішення завдань сайту. Веб-додаток входить до складу сайту, але веб-додаток без даних сайтом є тільки технічно. Оболонку (форму, шаблон) потрібно наповнити і активізувати. У більшості випадків в Інтернеті одному сайту відповідає одне доменне ім'я. Саме по доменних іменах сайти ідентифікуються в глобальній мережі. Можливі інші варіанти: один сайт на декількох доменах або кілька сайтів під одним доменом. Зазвичай кілька доменів використовують великі сайти (веб-портали), щоб логічно відокремити різні види послуг, що надаються (mail.google.com, news.google.com, maps.google.com). Непоодинокі й випадки виділення окремих доменів для різних країн або мов. Наприклад, google.ru та google.fr логічно є сайтом Google на різних мовах, але технічно це різні сайти.

Об'єднання декількох сайтів під одним доменом характерно для безкоштовних хостингів. Іноді для ідентифікації сайтів в адресі після вказівки хоста стоїть тильда і ім'я сайту: example.com/~my-site-name/ (пор. З / home), а найчастіше використовується домен третього рівня: my-site-name.example. Com.

1.2 Зберігання

Апаратні сервери для зберігання сайтів називаються веб-серверами. Сама послуга зберігання називається хостингом. Раніше кожен сайт зберігався на своєму власному сервері, але з ростом Інтернету, технологічним поліпшенням серверів на одному комп'ютері стало можливе розміщення безлічі сайтів (віртуальний хостинг). Зараз сервери для зберігання тільки одного сайту називаються виділеними (англ. Dedicated). Один і той же сайт може бути доступний за різними адресами і зберігатися на різних серверах. Копія оригінального сайту в такому випадку називається дзеркалом. Існує також поняття оффлайновая версія сайту - це копія сайту, яка може бути переглянута на будь-якому комп'ютері без підключення до комп'ютерної мережі і використання серверного програмного забезпечення (ПО). При розробці сайту його тестують і налагоджують саме в оффлайновой версії, для того, щоб не демонструвати нісенітницю і помилки, прорахунки великого проекту. Саме для тестування в корпоративній мережі, або на початку в Інтернеті з обмеженим доступом під паролем запрошуються досвідчені «тестери». Це дозволяє прискорити виробництво великих проектів і налагодити їх для масового відвідувача (користувача). Особливу роль виконують по розробці і обслуговуванню сайту (порталу) адміністратори (по-іншому – адміни, згідно інтернет-сленгу). Якщо виготовлення форми (оболонки) виконує група або дуже кваліфікований фахівець (програміст, веб-дизайнер, системний адміністратор (згідно інтернет-сленгу - системний

адміністратор), координатор, він же адміністратор проекту), то обслуговування та інформаційне наповнення сайту підпорядковане, як правило, стратегічним завданням, вирішенням яких займається вся команда учасників проекту під керуванням адміністратора проекту (сайту, порталу).

Зараз напрацьовано багато програм і «пісалок» в технології PHP, але це підвищило і вимоги до кваліфікації учасників проекту, в зв'язку з многопрофильностью вирішуваних завдань. Просто сторінка (сайт-візитка) може готуватися секретарем-референтом. Проекти великих сайтів і порталів можуть зробити тільки обізнані і зацікавлені фахівці. Активна комунікація на сайті (порталі) часто виконує функцію директора напрямку і офісу зі службою супроводу (листування, комутатор прямого спілкування, оперативна довідка, і ін.). Багато сайтів (порталів) оновлюють частіше ніж раз в день, а інтернет-магазини – за фактом руху товару (нові надходження, відсутності товару в наявності). Новинні сайти реально виставляють інформацію з точністю до хвилини, так як журналісти мають пріоритети на цитування першоджерел згідно з авторським правом, пріоритету посилань, рейтингу та ін.

Типи інтернет-ресурсів:

За доступності сервісів

- Відкриті – всі сервіси повністю доступні для будь-яких відвідувачів і користувачів.
- Напіввідкриті – для доступу необхідно зареєструватися (зазвичай безкоштовно).
- Закриті – повністю закриті службові сайти організацій (у тому числі корпоративні сайти), особисті сайти приватних осіб. Такі сайти доступні для вузького кола користувачів. Доступ новим користувачам зазвичай дається через так звані інвайт (запрошення).

За фізичним розташуванню:

- Загальнодоступні сайти мережі Інтернет.
- Локальні сайти – доступні тільки в межах локальної мережі. Це можуть бути як корпоративні сайти організацій, так і сайти приватних осіб в локальній мережі провайдера.

За схемою подання інформації, її обсягом і категорії вирішуваних завдань можна виділити наступні типи веб-ресурсів

- Інтернет-портал – багатокомпонентна розгалужена структура, скомпонована з функціонально самодостатніх сайтів самостійних організацій або підрозділів корпоративної структури.
- Тематичний сайт – сайт, який надає специфічну вузькотематичні інформацію по якій-небудь темі.
- Тематичний портал – це дуже великий веб-ресурс, який надає вичерпну інформацію з певної тематики. Портали схожі на тематичні сайти, але додатково містять засоби взаємодії з користувачами і дозволяють користувачам спілкуватися в рамках порталу (форуми, чати) – це середовище існування користувача.
- Сайт-візитка – містить загальні дані про власника сайту (організація або індивідуальний підприємець): вид діяльності, історія, прейскурант, контактні дані, реквізити, схема проїзду. Фахівці розміщують своє резюме (тобто детальна візитна картка).
- Представницький сайт – так іноді називають сайт-візитку з розширеною функціональністю: докладний опис послуг, портфоліо, відгуки, форма зворотного зв'язку і т. д.
- Корпоративний сайт – містить повну інформацію про компанії-власника, послуги / продукцію, події в житті компанії. Відрізняється від сайту-візитки і представницького сайту повнотою наданої

інформації, часто містить різні функціональні інструменти для роботи з контентом (пошук і фільтри, календарі подій, фотогалереї, корпоративні блоги, форуми). Може бути інтегрований з внутрішніми інформаційними системами компанії-власника (КІС, CRM, бухгалтерськими системами). Може містити закриті розділи для тих чи інших груп користувачів – співробітників, дилерів, контрагентів тощо.

- Каталог продукції – в каталозі є докладний опис товарів / послуг, сертифікати, технічні та споживчі дані, відгуки експертів і так далі. На таких сайтах розміщується інформація про товари / послуги, яку неможливо помістити в прейскурант.
- Інтернет-магазин – сайт з каталогом продукції, за допомогою якого клієнт може замовити потрібні йому товари. Використовуються різні системи розрахунків: від пересилання товарів післяплатою або автоматичною пересилання рахунку по факсу до розрахунків за допомогою пластикових карт.
- Промосайт – сайт про конкретну торгову марку або продукт, на таких сайтах розміщується вичерпна інформація про бренд, різних рекламних акціях (конкурси, вікторини, ігри і т. п.).
- Сайт-квест – інтернет-ресурс, на якому організовано змагання з розгадування послідовно взаємопов'язаних логічних загадок.

Технологічні особливості

За технологічними особливостями створення і відображення сайти розрізняються:

По технологии відображення

- Статичні – складаються з статичних html (htm, dhtml) сторінок, що становлять єдине ціле. Користувачеві видаються файли в тому вигляді, в якому вони зберігаються на сервері.

❖ Динамічні – складаються з динамічних html (htm, dhtml) сторінок-шаблонів, інформації, скриптів та іншого у вигляді окремих файлів.

Вміст генерується за запитом спеціальними скриптами (програмами) на основі інших даних з будь-якого джерела

❖ Сайти, створені із застосуванням т. Н. Flash-технологій, коли весь сайт розташовується на одній веб-сторінці, призначеної виключно для завантаження Flash-файлу, а вся навігація і контент реалізовані в самому Flash-ролику.

За типами макетів

❖ Фіксованої ширини (англ. Rigid fixed) – розміри елементів сторінки мають фіксоване значення, незалежне від дозволу, розміру, співвідношення сторін екрану монітора і розмірів вікна браузера, задається в абсолютних значеннях – PX.

❖ Гумовий макет (англ. Adaptable fluid) – розміри несучих елементів, значення ширини, задаються відносним значенням -% (відсотки), сторінки відображаються на весь екран монітора по ширині.

❖ Динамічно еластичний тип макета (англ. Dynamically expandable elastic) – розміри більшості елементів задаються відносними значеннями - EM і% (відсотки). Всі відносні пропорції розмірів елементів завжди залишаються незмінними, незалежно від дозволу, розміру, співвідношення сторін екрану монітора, розмірів вікна і масштабу вікна оглядача. І завжди постійні щодо вікна оглядача.

❖ Адаптивний (англ. Adaptive) – дизайн сторінки підлаштовується (адаптується) під розмір екрану, в тому числі може відбуватися перебудова блоків з одного місця на інше, або їх заміна блоками відображеними тільки при певному вирішенні.

Програмування:

Клієнтські мови

Як випливає з назви, програми на клієнтських мовах обробляються на стороні користувача, як правило, їх виконує браузер. Це і створює головну проблему клієнтських мов – результат виконання програми (скрипта) залежить від браузера користувача. Тобто, якщо користувач заборонив виконувати клієнтські програми, то вони виконуватися не будуть, хоч би як бажав цього програміст. Крім того, може статися таке, що в різних браузерах або в різних версіях одного і того ж браузера один і той же скрипт буде виконуватися по-різному. З іншого боку, якщо програміст покладає надії на серверні програми, то він може спростити їх роботу і знизити навантаження на сервер за рахунок програм, виконуваних на стороні клієнта, оскільки вони не завжди вимагають перезавантаження (генерацію) сторінки.

Серверні мови

Коли користувач дає запит на яку-небудь сторінку (переходить на неї по посиланню або вводить адресу в адресному рядку свого браузера), то викликана сторінка спочатку обробляється на сервері, тобто виконуються всі програми, пов'язані зі сторінкою, і тільки потім повертається до відвідувача по мережі у вигляді файлу. Цей файл може мати розширення HTML, PHP, ASP, ASPX, Perl, SSI, XML, DHTML, XHTML. Робота програм вже повністю залежна від сервера, на якому розташований сайт, і від того, яка версія тієї чи іншої мови підтримується. До серверним мов програмування можна віднести PHP, Perl, Python, Ruby, будь .NET мову програмування (технологія ASP.NET), Java, Groovy, Javascript. Важливою стороною роботи серверних мов є можливість організації безпосередньої взаємодії з системою управління базами даних (або СУБД) – сервером бази даних, в якій впорядковано зберігається інформація, яка може бути викликана в будь-який момент.

1.3 Безпека

Існує безліч сайтів, які є значущими ресурсами. На цих ресурсах можуть розташовуватися персональні дані користувачів (наприклад, особисте листування, адреси, телефони) або фінансова інформація (наприклад, банківські сайти). Злом таких ресурсів може спричинити як прямі грошові збитки (наприклад, зловмисник може перерахувати гроші з чужого рахунку на свій власний), так і непрямі, пов'язані з поширенням конфіденційної інформації або просто зловмисник може зіпсувати вміст сайту.

Для багатьох сайтів важливо забезпечити певний рівень безпеки. Необхідний рівень безпеки багато в чому залежить від церкви, розміщеної на сайті інформації.

Найбільш поширені прояви злому сайту:

Несанкціонованих змін зловмисниками відображення сайту (див .: дефейсінг, хакери) підробка сайту (дизайн і вміст сайту може бути скопійовано і в користувача такого сайту можуть вкрасти паролі) зниження числа користувачів сайту через злодійство користувачів, які перейшли на сайт з пошукової системи або мобільних пристроїв поява посилань на зовнішні ресурси (чорне seo) поява порно-банерів та іншої настирливої реклами.

Вторинні наслідки злому сайту:

Блокування сайту як «шкідливого» пошуковими системами Google і Яндекс блокування сайту браузерами Google Chrome, Opera, Яндекс. Браузер та іншими блокування сайту антивірусами блокування сайту хостинг-провайдером, на якому він розташований зниження позицій сайту в пошуковій видачі пошукових систем зниження кількості щоденних відвідувачів сайту Найбільш популярними мотивами злому сайту є: підірвати продажу або імідж конкуруючого сайту отримати вигоду:

розсилати за гроші спам з сайту; перенаправляти за гроші користувачів сайту на інші сайти і сторінки додатка Google Play і AppStore; використовувати сайт для DDoS-атак; використовувати сайт для розміщення на ньому посилань на зовнішні сайти; розміщення шкідливого коду, що заражає комп'ютери відвідувачів сайту шантаж: злодійство з метою повернення власнику за гроші реклама: розміщення на сайті дефейсінга з метою реклами хакерських послуг політичні мотиви: з метою показати позицію щодо того чи іншого політичного ладу або організації За даними, проведеного сервісом щодо захисту сайтів SiteSecure, дослідження безпеки комерційних сайтів в Росії за 1 квартал 2015 года кожен 10-й сайт заражений або має високий ризик зараження і блокування за шкідливість.

1.4 Висновки до розділу 1

Створення власного web-додатку надає будь-якій компанії низку нових можливостей, якщо процес його розробки було виконано з урахуванням особливостей предметної області та потреб замовника на кожному етапі: створення, опрацювання технічного завдання.

Також слід зазначити, що додаток приноситиме очікувані результати лише у випадку постійної актуалізації та відповідності сучасним тенденціям у web-розробці.

РОЗДІЛ 2.
ОСОБЛИВОСТІ РОЗРОБКИ WEB-ДОДАТКУ

Під web-додатком розуміється прикладна програма, розроблена з архітектури "клієнт-сервер", що використовує в якості клієнта web-браузер і працює на стороні web-сервера (Рис. 2.1. Загальна схема взаємодії користувача з web-додатком). При цьому взаємодія між клієнтом і сервером виконується з використанням протоколу HTTP.

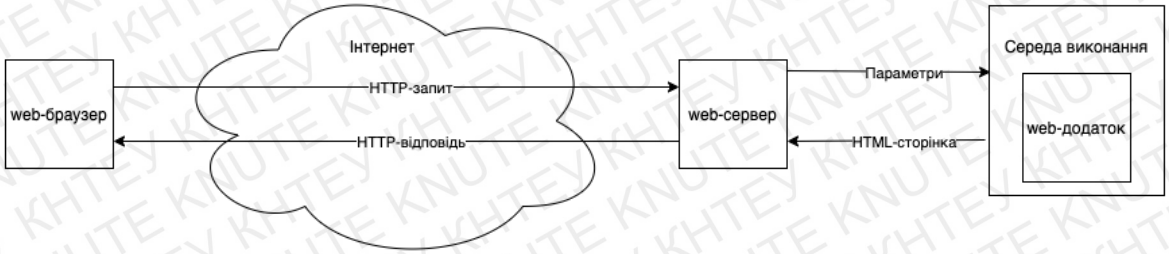

Рис. 2.1. Загальна схема взаємодії користувача з web-додатком

Веб-додаток может бути клієнтом інших служб, наприклад, бази даних або іншого веб-додатку, розташованого на іншому сервері.

Розробку сайту логістичних послуг я буду виконувати на мовах програмування PHP та JavaScript. Буду використовувати Open Server, системи керування базами даних SQLite та систему контролю версій Git.

2.1 Програмування на стороні сервера

PHP – одна з найпоширеніших мов програмування на сьогоднішній день. Започаткована у 1994, вона постійно покращюється командою розробників і має дуже велику спільноту прихильників по всьому світу.

Це скриптова, об'єктно-орієнтована, інтерпретуєма мова відкритого програмного забезпечення, що використовується для реалізації логіки веб-додатків на стороні сервера [1].

До переваг PHP можна віднести:

- для всіх найбільш поширених операційних систем (Windows, MacOS, Linux) є свої версії пакетів розробки на PHP, а це означає, що ви можете створювати веб-сайти на будь-який з цих операційних систем
- PHP може працювати у зв'язці з різними веб-серверами: Apache, Nginx, IIS
- простота і легкість освоєння. Як правило, вже маючи навіть невеликий досвід в програмуванні на PHP, можна створювати повноцінні веб-сайти
- PHP схожий на мову Сі, тому, знаючи Сі або однією з мов з Сі-подібним синтаксисом, буде простіше опанувати PHP
- PHP підтримує роботу з великою кількістю систем керування баз даних (MySQL, MSSQL, Oracle, Postgre, MongoDB та інші)
- поширеність хостингових послуг і їх дешевизна. Так як, зазвичай, хостингові компанії розміщують веб-сайти на PHP на веб-серверах Apache або Nginx, які працюють на одній з операційних систем сімейства Linux. І веб-сервери, і операційні системи на базі Linux безкоштовні, що знижує загальну вартість використання хостингу
- постійний розвиток. PHP продовжує розвиватися, виходять все нові версії, які несуть нові функції, адаптуючи мову програмування до нових викликів. І, як правило, перейти на нову версію не складає труднощів
- наявність фреймворків та бібліотек видкритого первинного коду, що дають змогу скоротити час розробки та оптимізувати процеси

❖ до теперішнього моменту поточної стабільної версією є PHP 7.4.

2.1.1 Symfony фреймворк

Symfony – відкритий PHP-фреймворк, що автоматизовує різноманітні веб-задачі, являє собою широконалаштовну систему пов'язаних класів і призначений для розробки та керування веб-застосунками.

Symfony спрямований на прискорення створення та підтримки веб-застосунків, а також для уникнення витрат часу для розв'язування тривіальних задач у розробці. За допомогою акселератора PHP Symfony збільшує продуктивність та зменшує навантаження на сервер. Також Symfony ставить за мету дати розробникам повний контроль над конфігурацією: майже все можливо налаштувати, від структури каталогів до сторонніх бібліотек.

Ще однією перевагою Symfony є наявність генераторів, за допомогою яких значно пришвидшується розробка. Підтримує велику кількість баз даних, серед яких MySQL, Oracle, PostgreSQL, SQLite, Microsoft SQL Server, MongoDB тощо.

Серед можливостей: інструменти для локалізації та інтернаціоналізації, unit-тестування, БД-абстракції, smart-URL, Debug Toolbar, development та production режими, form framework і, навіть, пропонує власний веб-сервер.

2.1.2 Laravel фреймворк

Ще один безкоштовний, з відкритим кодом PHP-фреймворк, створений і призначений для розробки веб-додатків відповідно до шаблону model–view–controller (MVC). Має багато спільного з Symfony, навіть використовую звідти деякі компоненти. Однак, в свою чергу, має деякі фундаментальні відмінності у реалізації, що робить Laravel простішим і значно знижує поріг входу у програмування промислового коду.

2.1.3 Model-View-Controller

Модель – вигляд – контролер (англ. Model-view-controller, MVC) – архітектурний шаблон, який використовується під час проектування та розробки програмного забезпечення.

Цей шаблон передбачає поділ системи на три взаємопов'язані частини: модель даних, вигляд (інтерфейс користувача) та модуль керування. Застосовується для відокремлення даних (моделі) від інтерфейсу користувача (вигляду) так, щоб зміни інтерфейсу користувача мінімально впливали на роботу з даними, а зміни в моделі даних могли здійснюватися без змін інтерфейсу користувача.

Мета шаблону – гнучкий дизайн програмного забезпечення, який повинен полегшувати подальші зміни чи розширення програм, а також надавати можливість повторного використання окремих компонентів програми. Крім того використання цього шаблону у великих системах сприяє впорядкованості їхньої структури і робить їх більш зрозумілими за рахунок зменшення складності.

У рамках архітектурного шаблону модель – вигляд – контролер (MVC) програма поділяється на три окремі, але взаємопов'язані частини з розподілом функцій між компонентами. Модель (Model, або Entity у деяких випадках) відповідає за зберігання даних та їх структуру. Вигляд (View) відповідальний за представлення цих даних користувачеві, тобто інтерфейс програми. Контролер (Controller) керує компонентами, отримує сигнали у вигляді реакції на дії користувача (зміна положення курсора миші, натискання кнопки, ввід даних в текстове поле) і передає дані у модель.

- Модель є центральним компонентом шаблону MVC і відображає поведінку застосунку, незалежну від інтерфейсу користувача. Модель

стосується прямого керування даними, логікою та правилами застосунку.

- Вигляд може являти собою будь-яке представлення інформації, одержуване на виході, наприклад графік, діаграму чи html-сторінку. Одночасно можуть співіснувати кілька виглядів (представлень) однієї і тієї ж інформації, наприклад гістограма для керівництва компанії й таблиці для бухгалтерії.
- Контролер одержує вхідні дані й перетворює їх на команди для моделі чи вигляду.

Модель інкапсулює ядро даних та основний функціонал їхньої обробки і не залежить від процесу вводу чи виводу даних.

Вигляд може мати декілька взаємопов'язаних областей, наприклад різні таблиці і поля форм, в яких відображаються дані.

У функції контролера входить відстеження визначених подій, що виникають в результаті дій користувача. Контролер дозволяє структурувати код шляхом групування пов'язаних дій в окремий клас. Наприклад у типовому MVC-проекті може бути користувацький контролер, що містить групу методів, пов'язаних з управлінням обліковим записом користувача, таких як реєстрація, авторизація, редагування профілю та зміна пароля.

Зареєстровані події транслюються в різні запити, що спрямовуються компонентам моделі або об'єктам, відповідальним за відображення даних.

Відокремлення моделі від вигляду даних дозволяє незалежно використовувати різні компоненти для відображення інформації. Таким чином, якщо користувач через контролер внесе зміни до моделі даних, то інформація, подана одним або декількома візуальними компонентами, буде автоматично відкоригована відповідно до змін, що відбулися.

2.2 Програмування на стороні клієнта

JavaScript класифікують як прототипну (підмножина об'єктно-орієнтованої), скриптову мову програмування з динамічною типізацією. Окрім прототипної, JavaScript також частково підтримує інші парадигми програмування (імперативну та частково функціональну) і деякі відповідні архітектурні властивості, зокрема: динамічна та слабка типізація, автоматичне керування пам'яттю, прототипне наслідування, функції як об'єкти першого класу [3].

Мова JavaScript використовується для:

- написання сценаріїв веб-сторінок для надання їм інтерактивності
- створення односторінкових веб-застосунків (React, AngularJS, Vue.js)
- програмування на стороні сервера (Node.js)
- стаціонарних застосунків (Electron, NW.js)
- мобільних застосунків (React Native, Cordova)
- сценаріїв в прикладному ПЗ (наприклад, в програмах зі складу Adobe Creative Suite чи Apache JMeter)
- всередині PDF-документів тощо

Сьогодні JavaScript є, напевне, самою популярною мовою серед молодого покоління. Стрімке зростання сфер застосування та вихід нових, зручних, бібліотек та фреймворків привернуло величезну увагу серед розробників і зацікавило навіть тих, хто скептично ставився для до цієї мови. Зараз ні один сайт не обходиться без застосування JavaScript.

2.2.1 Бібліотека jQuery

jQuery — популярна JavaScript-бібліотека з відкритим кодом. Згідно з дослідженнями організації W3Techs, JQuery використовується понад половиною від мільйона найвідвідуваніших сайтів. jQuery є

найпопулярнішою бібліотекою JavaScript, яка посилено використовується на сьогоднішній день.

Синтаксис jQuery розроблений, щоб зробити орієнтування у навігації зручнішим завдяки вибору елементів DOM, створенню анімації, обробки подій, і розробки AJAX-застосунків. jQuery також надає можливості для розробників, для створення плагінів у верхній частині бібліотеки JavaScript. Використовуючи ці об'єкти, розробники можуть створювати абстракції для низькорівневої взаємодії та створювати анімацію для ефектів високого рівня. Це сприяє створенню потужних і динамічних веб-сторінок.

2.2.2 Vue.js

Vue.js – це JavaScript-фреймворк що використовує шаблон MVVM для створення інтерфейсів користувача на основі моделей даних, через реактивне зв'язування даних.

MVVM (Model-View-ViewModel) – це шаблон проєктування, що застосовується під час проєктування архітектури застосунків (додатків). MVVM орієнтований на такі сучасні платформи розробки, як Windows Presentation Foundation та Silverlight від компанії Microsoft.

MVVM полегшує відокремлення розробки графічного інтерфейсу від розробки бізнес логіки (бек-енд логіки), відомої як модель (можна також сказати, що це відокремлення представлення від моделі). Модель представлення є частиною, яка відповідає за перетворення даних для їх подальшої підтримки і використання. З цієї точки зору, модель представлення більше схожа на модель, ніж на представлення і оброблює більшість, якщо не всю, логіку відображення даних. Модель представлення може також реалізовувати патерн медіатор, організовуючи доступ до бек-енд

логіки навколо множини правил використання, які підтримуються представленням.

MVVM використовується для відокремлення моделі та її відображення. Необхідністю цього є надання можливості змінювати їх незалежно одну від одної. Наприклад, розробник працює над логікою роботи з даними, а дизайнер – з користувацьким інтерфейсом.

Шаблон MVVM ділиться на три частини:

- модель (Model), як і в класичному шаблоні MVC, Модель являє собою фундаментальні дані, що необхідні для роботи застосунку
- вид/(Вигляд) (View) як і в класичному шаблоні MVC, Вигляд – це графічний інтерфейс, тобто вікно, кнопки тощо
- модель вигляду (ViewModel, що означає «Model of View») з одного боку є абстракцією Вигляду, а з іншого надає обгортку даних з Моделі, які мають зв'язуватись. Тобто вона містить Модель, яка перетворена до Вигляду, а також містить у собі команди, якими може скористатися Вигляд для впливу на Модель

Фактично ViewModel призначена для того, щоб:

- здійснювати зв'язок між моделлю та вікном
- відслідковувати зміни в даних, що зроблені користувачем
- відпрацьовувати логіку роботи View (механізм команд)

2.3 Система керування базами даних

Централізований характер управління даними в базі даних передбачає необхідність існування деякої особи (групи осіб), на яку покладаються функції адміністрування даними, що зберігаються в базі.

Система керування базами даних (СУБД) – це комплекс програмних і мовних засобів, необхідних для створення баз даних, підтримання їх в актуальному стані та організації пошуку в них необхідної інформації.

2.3.1 MySQL

MySQL – вільна система управління базами даних. MySQL є власністю компанії Oracle Corporation, що отримала її разом з поглиненою Sun Microsystems, яка здійснює розробку і підтримку додатку.

Цю систему управління базами даних з відкритим кодом було створено як альтернатива комерційним системам. MySQL із самого початку була дуже схожою на mSQL, проте з часом вона все розширювалася і зараз MySQL - одна з найпоширеніших систем управління базами даних. Вона використовується, у першу чергу, для створення динамічних веб-сторінок, оскільки має чудову підтримку з боку різноманітних мов програмування.
MySql є рішенням для малих і середніх додатків. Зазвичай використовується як сервер, до якого звертаються локальні або віддалені клієнти, проте до дистрибутиву входить бібліотека внутрішнього сервера, що дозволяє включати MySQL до автономних програм. Вихідні коди сервера компілюються на багатьох платформах. Найповніше можливості сервера виявляються в UNIX-системах.

Гнучкість СУБД MySQL забезпечується підтримкою великої кількості типів таблиць: користувачі можуть вибрати як таблиці типу MyISAM, що підтримують повнотекстовий пошук, так і таблиці InnoDB, що підтримують транзакції на рівні окремих записів. Більш того, СУБД MySQL поставляється із спеціальним типом таблиць EXAMPLE, що демонструє принципи створення нових типів таблиць. Завдяки відкритій архітектурі й GPL-ліцензуванню, в СУБД MySQL постійно з'являються нові типи таблиць.

MySQL характеризується великою швидкістю, стійкістю і простотою використання.

Для некомерційного використання MySQL є безкоштовною. Можливості сервера MySQL:

- простота у встановленні та використанні
- підтримується необмежена кількість користувачів, що одночасно працюють із БД
- кількість рядків у таблицях може досягати 50 млн.
- висока швидкість виконання команд
- наявність простої та ефективної системи безпеки

2.3.2 SQLite

SQLite – полегшена реляційна система керування базами даних. Втілена у вигляді бібліотеки. Сирцевий код SQLite поширюється як суспільне надбання, тобто може використовуватися без обмежень та безоплатно. Фінансову підтримку розробників SQLite здійснює спеціально створений консорціум, до якого входять такі компанії, як Adobe, Oracle, Mozilla, Nokia, тощо.

Особливістю SQLite є те, що вона не використовує парадигму клієнт-сервер, тобто рушій SQLite не є окремим процесом, з яким взаємодіє застосунок, а надає бібліотеку, з якою програма компілюється і рушій стає складовою частиною програми. Таким чином, як протокол обміну використовуються виклики функцій бібліотеки SQLite. Такий підхід зменшує накладні витрати, час відгуку і спрощує програму. SQLite зберігає всю базу даних (включаючи визначення, таблиці, індекси і дані) в єдиному стандартному файлі на тому комп'ютері, на якому виконується застосунок.

Простота реалізації досягається за рахунок того, що перед початком виконання транзакції весь файл, що зберігає базу даних, блокується, а ACID-функції досягаються зокрема за рахунок створення файлу-журналу.

Кілька процесів або потоків можуть одночасно без жодних проблем читати дані з однієї бази. Запис в базу можна здійснити тільки в тому випадку, коли жодних інших запитів у цей час не обслуговується, інакше спроба запису закінчується невдачею, і в програму повертається код помилки. Іншим варіантом розвитку подій є автоматичне повторення спроб запису протягом заданого інтервалу часу.

Таким чином SQLite є гарним варіантом для невеликих проектів, або проектів без нагальної необхідності в окремому сервері баз даних.

Ще одне поняття, про яке треба говорити в контексті баз даних – це ORM, технологія програмування, яка зв'язує бази даних з концепціями об'єктно-орієнтованих мов програмування, створюючи “віртуальну об'єктну базу даних”. Суть проблеми полягає в перетворенні таких об'єктів у форму, в якій вони можуть бути збережені у файлах або базах даних, і які легко можуть бути витягнуті в подальшому, зі збереженням властивостей об'єктів і відношень між ними.

ORM покликана позбавити програміста від написання великої кількості коду, часто одноманітного і схильного до помилок, тим самим значно підвищуючи швидкість розробки. Крім того, більшість сучасних реалізацій ORM дозволяє програмістові при необхідності жорстко задати код SQL-запитів, який використовуватиметься при тих чи інших діях (збереження в базу даних, завантаження, пошук тощо) з постійним об'єктом.

2.4 Open Server Panel

Open Server це портативна серверна платформа і програмне середовище, створене спеціально для веб-розробників, використовуючих операційну систему Windows.

Програмний комплекс має багатий набір серверного програмного забезпечення, зручний, багатофункціональний продуманий інтерфейс, має потужні можливості з адміністрування та налаштування компонентів. Платформа широко використовується з метою розробки, налагодження і тестування веб-проектів, а так само для надання веб-сервісів в локальних мережах.

Переваги Open Server:

- портативність системі – може бути запущений навіть зі стороннього накопичувача.
- легкість в установці
- велика кількість як автоматизованих налаштувань, так і повноцінні можливості через термінал
- можливість швидко перемикати версії php, системи керування базами даних, налаштування сервера

2.5 Composer

Composer – це менеджер пакетів прикладного рівня для мови програмування PHP що забезпечує стандартний формат для управління залежностями у програмному забезпеченні та необхідними бібліотеками.

Composer працює з командного рядка і встановлює залежності (наприклад, бібліотек) для застосунку. Він також дозволяє користувачам встановлювати PHP пакети, доступні за ресурсом «Packagist», який є його основним сховищем, яке містить доступні пакети. Він також реалізує

автозавантажувач класів, для встановлених бібліотек і це полегшує використання коду від сторонніх розробників.

Сьогодні Composer застосовується при розробці більшості web-проектів, також використовується як складова частина декількох популярних PHP проектів з відкритим вихідним кодом, наприклад: Laravel, Symfony.

2.6 Висновки до розділу2

Таким чином SQLite є гарним варіантом для невеликих проектів, або проектів без нагальної необхідності в окремому сервері баз даних. Система керування базами даних (СУБД) – це комплекс програмних і мовних засобів, необхідних для створення баз даних, підтримання їх в актуальному стані та організації пошуку в них необхідної інформації.

РОЗДІЛ 3.
РОЗРОБКА САЙТУ

Для розробки сайту автор використав фреймворк Symfony 5. Це прискорить розробку та дасть можливість використовувати різні бібліотеки. Сайт буде мати три сторінки:

1. головна сторінку з інформацією про компанію
2. галерея автомобілів, з якої можна буде перейти на сторінку кожного авто та подивитись його характеристики
3. сторінка с калькулятором вартості автомобіля

Вартість автомобіля буде вираховуватись на основі введеної заказником інформації.

Для початку треба встановити інсталятор Symfony. Це бінарний файл для спрощеної комунікації з фреймворком. Встановлюємо за допомогою CURL:

curl -sS https://get.symfony.com/cli/installer | bash

Далі створюємо проект, в мене він має назву “carsDelivery”:

symfony new carsDelivery --version=5.0

Після цього Composer звітує, що проект створено: “Creating a new Symfony 5.0 project with Composer; Your project is now ready” і ми вже можемо бачити структуру проекту (Рис. 3.1. Структура проекту).

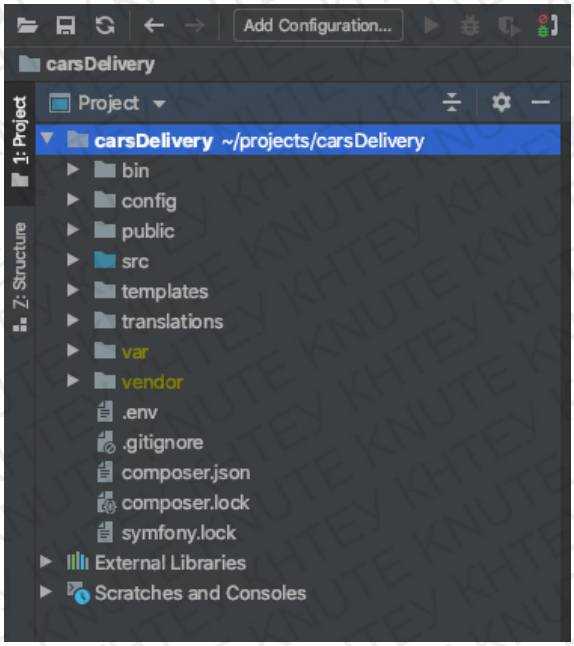

Рис. 3.1. Структура проекту

3.1 Створення моделі

Першим кроком треба задати значення для змінних “APP_ENV” та “DATABASE_URL” у файлі конфігураційному файлі “.env”. Перша змінна відповідає за навколишнє середовище (може бути тестовим або робочим), а друга вказує шлях до бази даних.

Використовуєм SQLite тому, що для цього не потрібен сервер і не потрібно витрачати ресурси комп’ютера. Для роботи с базою даних в Symfony використовується ORM під назвою Doctrine.

Встановлюємо ORM командою:

composer require symfony/orm-pack

Потрібно казати Doctrine яку СКБД використовувати. Для цього заповнимо конфігурації Doctrine (Рис. 3.2. Конфігурація Doctrine). Після цього можна створювати модель згідно MVC.

В Symfony модель називається Entity. Це об’єктно-орієнтована модель запису в базі. Це дає змогу, вибираючи один запис з бази, працювати з ним як з повноцінним об’єктом.

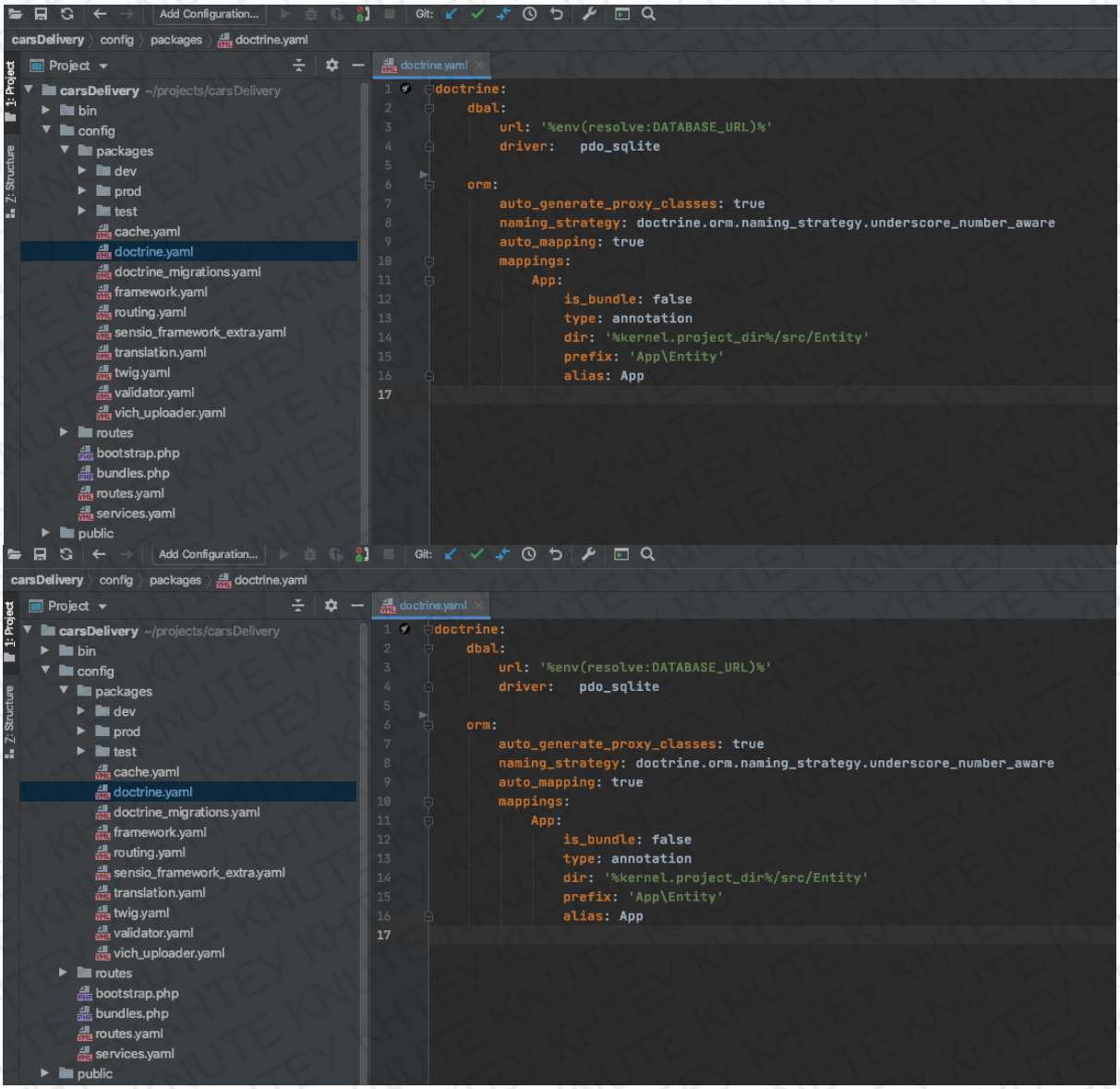

Рис. 3.2. Конфігурація Doctrine

Так як логістичної компанії займається автомобілями, то і головна сутність сайту буде автомобіль. Окремою сутністю буде країна з вартістю доставки автомобіля звідти. Також СКБД повинна сама автоматично створити кілька таблиць для зберігання внутрішньої інформації.

В результаті отримаємо схему простої бази даних (Рис. 3.3. Зображення схеми бази даних).

Після того, як Entity створені, потрібно запустити команду на створення бази даних та оновлення схеми.

php bin/console doctrine:database:create

php bin/console doctrine:schema:update --force

Після чого ми можемо бачити звіт в консолі, що схема оновлена: “[OK] Database schema updated successfully!”

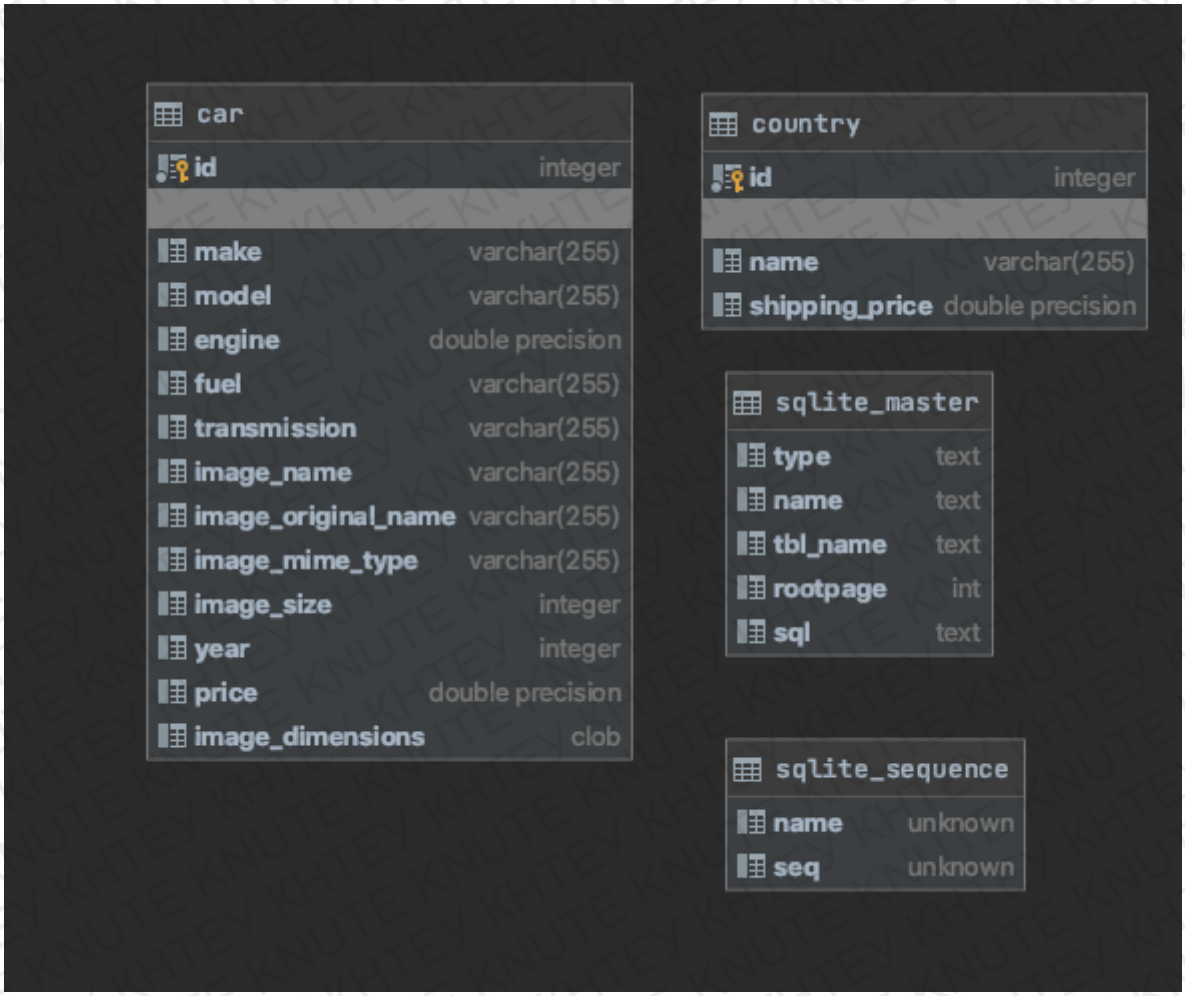

Рис. 3.3. Зображення схеми бази даних

3.2 Створення контроллера

Створимо контроллер для відображення галереї автомобілів (Рис. 3.4. Зображення контроллера для відображення галереї). Перш за все треба задати правило, що всі роути у даному контролеру починаються мають однаковий префікс “/car”. Тому для метода index() у якості роута достатньо буде лише одного слеша.

Успадкування від AbstractController надає нам, в свою чергу через ін'єкцію залежностей, доступ до самої Doctrine та методів з вже реалізованим поверненням об’єкта типу Response.

Також для коректної роботи галереї нам необхідно реалізувати пагінацію для виведення великої кількості зображень з автомобілями.

CarController.php

```php
<?php

namespace App\Controller;

use App\Entity\Car;
use App\Form\CarType;
use App\Repository\CarRepository;
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;

/**
 * @Route("/car")
 */
class CarController extends AbstractController
{
    private const LIMIT_PER_PAGE = 6;

    /**
     * @Route("/", name="car_index", methods={"GET"})
     */
    public function index(CarRepository $carRepository, Request $request): Response
    {
        $offset = $request->query->getInt( key: 'offset',  default: 0);
        $cars = $carRepository->getListWithPagination( limit: self::LIMIT_PER_PAGE, $offset);
        $total = $carRepository->count([]);

        return $this->render(
            view: 'car/index',
            [
                'cars' => $cars,
                'total' => $total,
                'offset' => $offset,
                'limit' => self::LIMIT_PER_PAGE,
            ]
        );
    }
```

3.4. Зображення контроллера для відображення галереї

Щоб реалізувати пагінацію задамо константу LIMIT_PER_PAGE, в яку запишему максимальну кількість сутностей, яка буде відображатися одночасно на сторінці.

Є два шляхи рахування пагінації – надсилати номер сторінки і рахувати відступ на стороні сервера, або робити це на стороні клієнта і надсилати на сервер одразу відступ. Реалізуємо другий варіант.

Будемо вважати, що в запиті буде надходити параметр “offset”. Якщо такого немає – за замовчуванням рахуємо, що відступ дорівнює нулю. Далі викликаємо метод з репозиторія для витягу результатів з бази (Рис. 3.5. Репозиторій для роботи за базою даних).

```php
<?php

namespace App\Repository;

use ...

/**
 * @method Car|null find($id, $lockMode = null, $lockVersion = null)
 * @method Car|null findOneBy(array $criteria, array $orderBy = null)
 * @method Car[]    findAll()
 * @method Car[]    findBy(array $criteria, array $orderBy = null, $limit = null, $offset = null)
 */
class CarRepository extends ServiceEntityRepository
{
    public function __construct(ManagerRegistry $registry)
    {
        parent::__construct($registry, entityClass: Car::class);
    }

    public function getListWithPagination(int $limit = 1, int $offset = 0): array
    {
        return $this->createQueryBuilder( alias: 'c')
            ->setMaxResults($limit)
            ->setFirstResult($offset)
            ->getQuery()
            ->getResult()
            ;
    }
}
```

Рис. 3.5. Репозиторій для роботи за базою даних

Цей метод приймає параметрами ліміт (для цього в контролері ми створили константу) і відступ, з якими потрібно зробити вибір з бази. Далі, щоб мати змогу оперувати цими даними в момент генерації HTML-сторінки, ми передаємо отримані результати у метод render(). Далі спираючись на ці дані мі відображаємо усю інформацію на сторінці (Рис. 3.5. Відображення вмісту галереї) Також на основі цих параметрів приймається рішення показувати кнопки переходу вперед-назад по галереї. Так, до прикладу, якщо це остання сторінка, кнопка “вправо” не буде з’являтися. В такому випадку повинна бути лише кнопка “вліво” (Рис.3.6. Остання сторінка галереї).

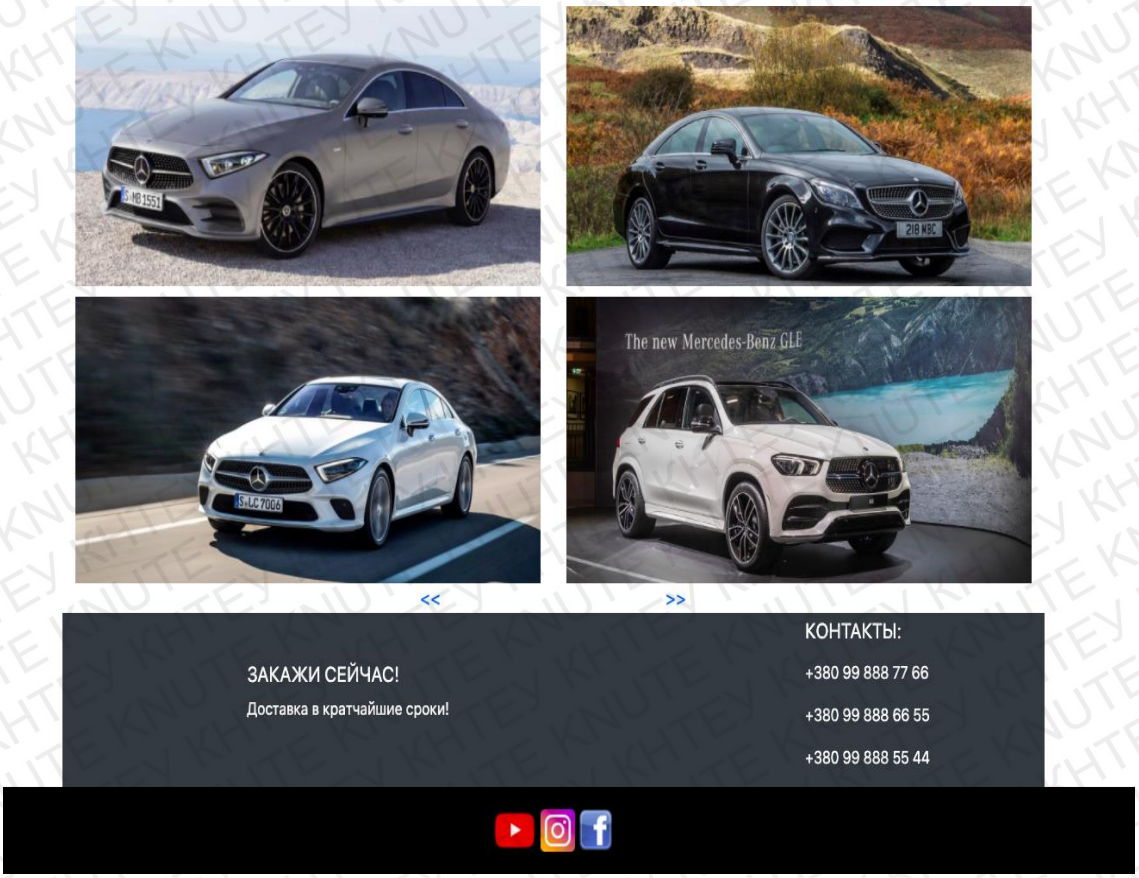

Рис. 3.5. Відображення вмісту галереї

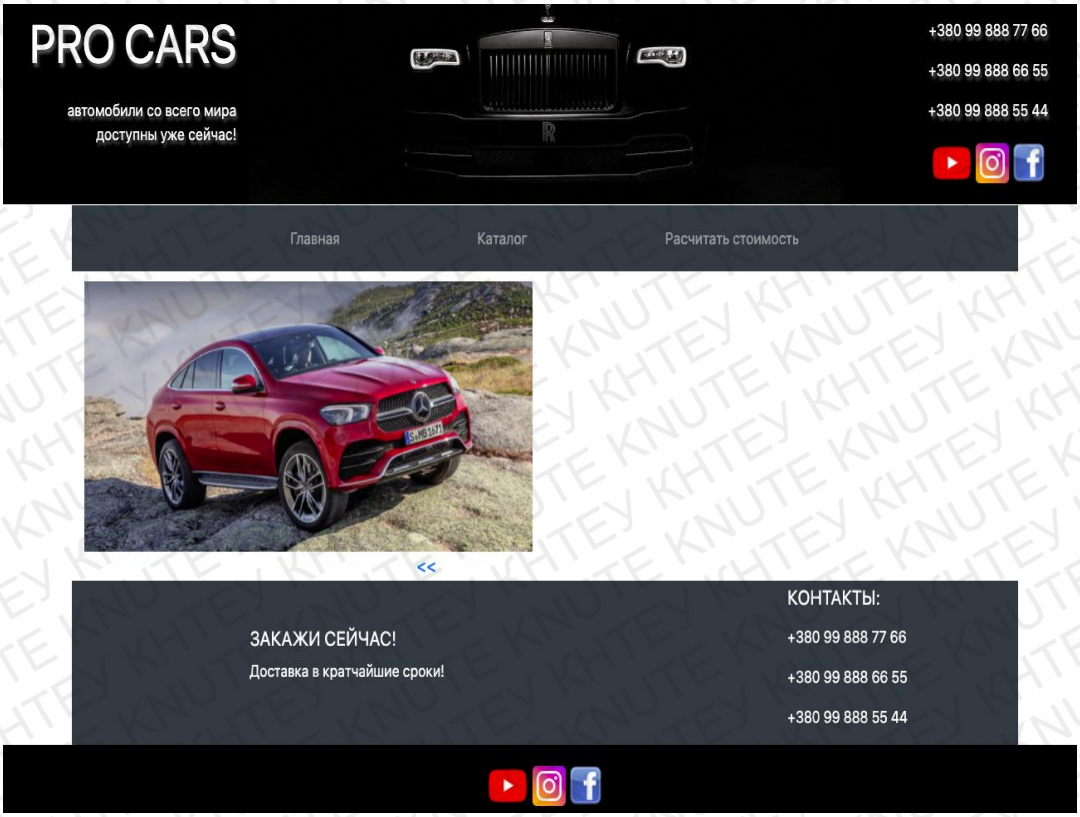

Рис. 3.6. Остання сторінка галереї

3.3 Створення сторінки калькулятора вартості

Наша логістична компанія займається доставкою автомобілів: тобто представник компанії знаходить, купує автомобіль в іншій країні і доставляє в Україну. Кінцева вартість для клієнта складається з суми початкової вартості авто, усіх податків, та вартості самої послуги. Тому треба зробити калькулятор який би міг розраховувати остаточну вартість автомобіля.

Для початку зробимо контроллер з роутом. Це досить просто, за виключенням одного моменту. На сторінці з калькулятором нам потрібен перечень країн з якими ми співпрацюємо, тобто країни, які є в нашій базі даних. Для цього створимо контроллер DeliveryController, обов'язково вкажемо роут (шлях за яким цей контроллер буде доступний для зовнішнього виклику) та передамо параметром репозиторій для зв'язку з таблицею, що зберігає країни. Далі звертаємось до репозиторія з проханням видати список

усіх країн та передаємо його далі з ключем “countries” на відображення (Рис. 3.7. Контроллер для сторінки калькулятора). Далі, на html-сторінці, ми можемо використати масив сутностей, що передали раніше і працювати с сутностями у контексті об’єктного програмування, звертаючись до їх властивостей (Рис. 3.8. Використання параметрів на html сторінці).

Щоб порахувати повну суму, яку повинен буде сплатити клієнт треба скласти первісну вартість автомобіля, вартість податків та вартість послуг нашої компанії.

Сума податків вираховується з суми акцизного збору, мита, збір в пенсійних фонд та ндс. В свою чергу ндс – це 20 процентів від суми початкової вартості авто, акцизного збору та мита.

Розрахування акцизного податку являє собою множину з базової ставки, коефіцієнту двигуна та коефіцієнту віку.

```php
<?php

namespace App\Controller;

use App\Repository\CountryRepository;
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;

class DeliveryCostController extends AbstractController
{
    /**
     * @Route("/delivery", name="delivery-cost")
     * @return Response
     */
    public function index(CountryRepository $countryRepository): Response
    {
        $countries = $countryRepository->findAll();

        return $this->render( view: 'delivery_cost', ['countries'=> $countries]);
    }
}
```

Рис. 3.7. Контроллер для сторінки калькулятора

```
<div class="row" style="...">
    <div class="col-6">Выберите страну</div>
    <div class="col-6">
        <div style="...">
            <select class="form-control" id="country">
                {% for country in countries %}
                    <option value='{{ country.shippingPrice }}'>{{ country.name }}</option>
                {% endfor %}
            </select>

        </div>
    </div>
</div>
```

Рис. 3.8. Використання параметрів на html сторінці

Якщо авто з бензиновим двигуном до 3 000 куб. см – базова ставка дорівнює 50 євро якщо об’єм двигуна понад 3000 куб см – 100 євро, для авто на дизелі з об'ємом двигуна до 3 500 куб см – 75 євро, понад 3 500 куб см – 150 євро.

Коефіцієнт двигуна – це одна тисячна частина від його об’єму. Коефіцієнт віку – це кількість років автомобілю.

Таким чином ми маємо череду простих арифметичних дій, яку треба лише реалізувати.

Реалізовуємо розрахунок вартості за допомогою JavaScript. Це дасть змогу користувачеві не чекати перезавантаження сторінки і робити все “на льоту”.

В першу чергу треба подбати про коректність даних. Тому при натисканні кнопки калькулятора треба перевіряти кожне поле вводу на наявність введених даних та на їх типізовану відповідність. У разі, якщо дані не введені, або введені некоректно, буде викинуто виключення у вигляді вікна з проханням ввести коректну інформацію (Рис. 3.9. Вікно з проханням ввести дані).

За допомогою jQuery шукаємо поля вводу по їх ідентифікаторах і перевіряємо одразу і на наявність введення даних, і завдяки функцій приведення типу parseInt та parseFloat, ще й на відповідність до типу (Рис. 3.10. Перевірка на введення коректних даних).

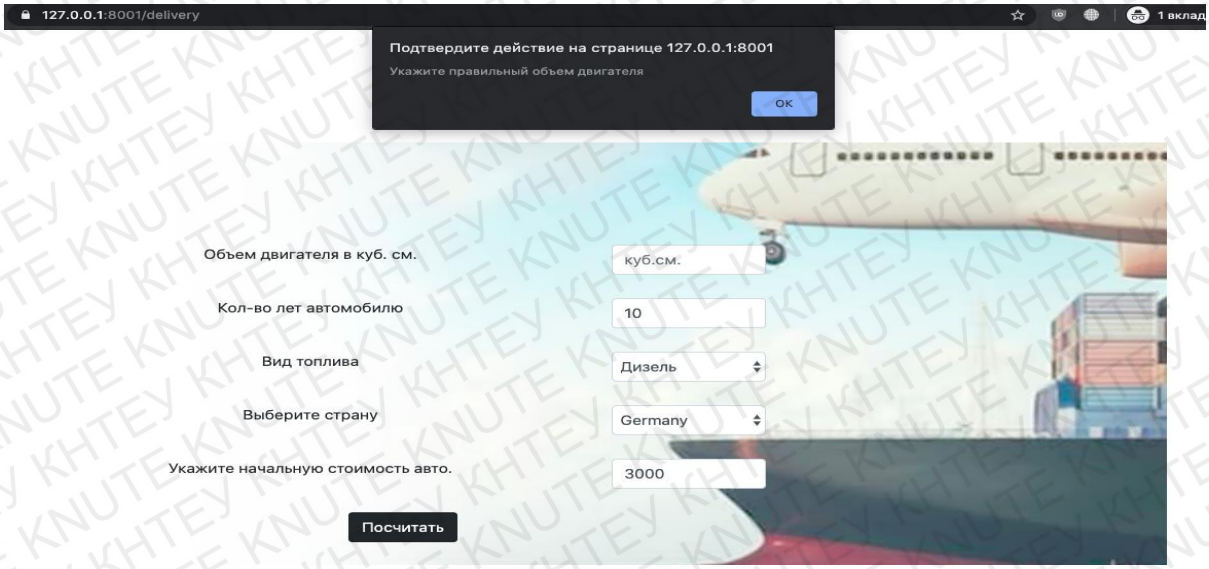

Рис. 3.9. Вікно з проханням ввести дані

```
<script type="text/javascript">

    $(function () {
        $('#calculate').on('click', function (e) {
            let engineSize = parseFloat($('#engine').val());
            if (!engineSize || isNaN(engineSize)) {
                alert("Укажите правильный объем двигателя");
                return;
            }
            let yearsCount = parseInt($('#years').val());
            if (!yearsCount || isNaN(engineSize)) {
                alert("Укажите сколько лет автомобилю");
                return;
            } else if (yearsCount === 0) {
                yearsCount = 1
            }
            let fuel = $("#fuel option:selected").val();
            if (!fuel) {
                alert("Укажите вид топлива");
                return;
            }
            let countryShippingPrice = parseInt($('#country').val());
            if (!countryShippingPrice || isNaN(countryShippingPrice)) {
                alert("Укажите страну");
                return;
            }
            let startPrice = parseFloat($('#startPrice').val());
            if (!startPrice || isNaN(startPrice)) {
                alert("Укажите правильный объем двигателя");
                return;
            }
```

Рис.3.10. Перевірка на введення коректних даних

Також за допомогою jQuery будемо рахувати вартість автомобіля. В окрему функцію винесемо розрахунки всіх податків getTaxAmount() (Рис 3.11.Функції розрахунків податків). Також відділимо розрахунок акцизу getAxcisTaxAmount(). Виклик основної функції буде підставляти результат у строку та відображати користувачу (Рис. 3.12. Результат розрахунків вартості автомобіля).

```
function getTaxAmount(engineSize, countYears, fuelType, startPrice) {
    const ndsPercent = 20;
    const tollPercent = 10;             //poshlinaya
    const retirementFeePercent = 5;     //pensionniy

    const axcisTax = getAxcisTaxAmount(engineSize, countYears, fuelType, startPrice,);
    const tollTax = startPrice / 100 * tollPercent;
    const ndsTax = (startPrice + axcisTax + tollTax) / 100 * ndsPercent;
    const retirementFeeTax = startPrice / 100 * retirementFeePercent;

    return parseFloat(axcisTax) + parseFloat(tollTax) + parseFloat(ndsTax) + parseFloat(retirementFeeTax);
}

function getAxcisTaxAmount(engineSize, countYears, fuelType) {
    let engineTax = 0;
    if (fuelType == 'petrol') {
        engineTax = engineSize < 3000 ? 50 : 100;
    } else if (fuelType == 'diesel') {
        engineTax = engineSize < 3000 ? 75 : 150;
    }
    const engineCoefficient = engineSize / 1000;
    const returnCoefficient = countYears;

    return engineTax * engineCoefficient * returnCoefficient;
}
```

(Рис 3.11.Функції розрахунків податків)

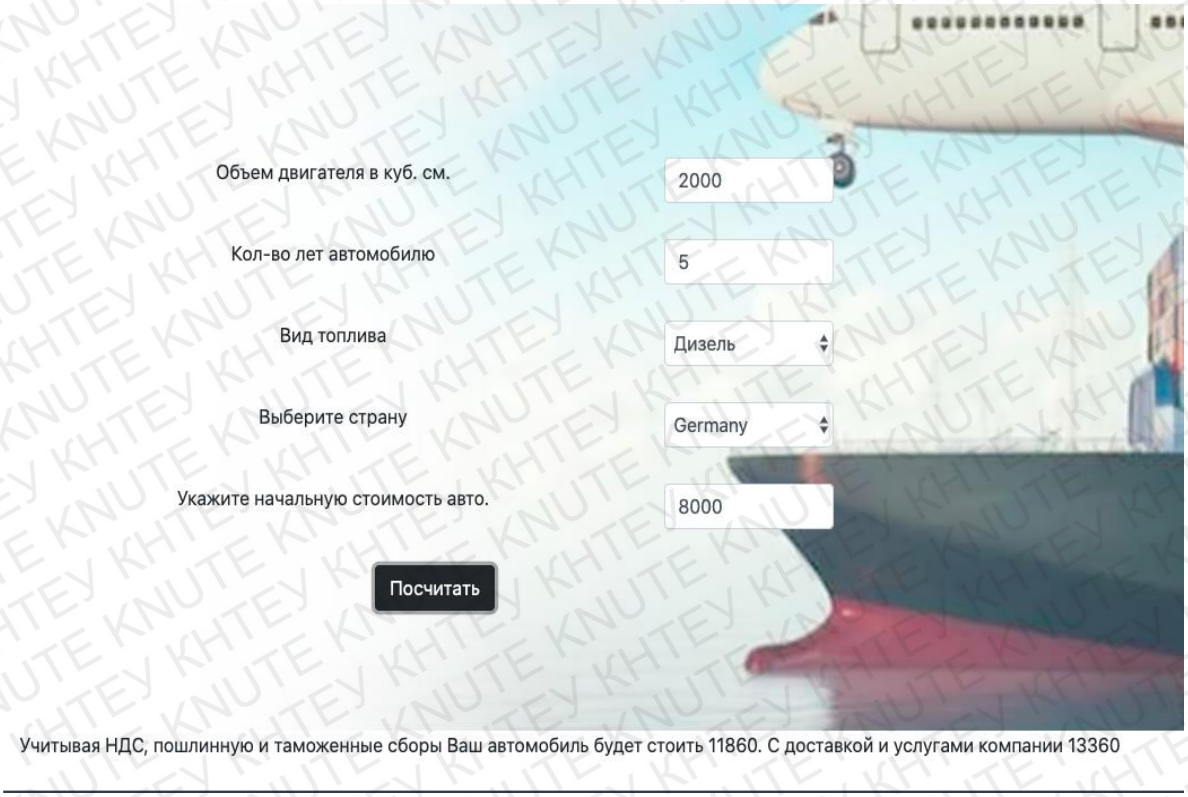

Рис. 3.12. Результат розрахунків вартості автомобіля

3.4 Висновки до розділу 3

В ході написання розділу, вказано на швидкодійність та зручність користування сайтом. Створення його так візуальне оформлення теж не меньше важливе для того щоби клієнту було зручніше орієнтуватись на сайті. Також простий і чіткий калькулятор вартості за послуги.

ВИСНОВКИ ТА ПРОПОЗИЦІЇ

У ході виконаного дослідження було визначено сутність поняття web-сайт, сформульовано перелік переваг наявності власного web-сайту для бізнесу.

Використвуючи такий сайт можно зручно та швидко керувати своїм бізнесом, або тимчасовими перевезеннями за допомогою переліку машин у каталозі сайту.

Програмувати подібні сайти зі схожими розрахунками та використовувати у своїх цілях. Створення яких не займає дуже багато часу і великих знань.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

Основний

1. Томсон Л., Веллинг Л. Разработка Web-приложений на PHP и MySql. ДиаСофтЮП, 2013. – 672с.
2. Веллинг Л., Томсон Л. Разработка веб-приложений с помощью PHP и MySQL. – М.: Вильямс - 2014. - 848 с.
3. Дейт К. Дж. Введение в системы баз данных = Introduction to Database Systems. – 8-е изд. / Дейт К. Дж. Вильямс, 2005. – 1328 с.
4. Кузнецов С. Д. Основы баз данных. – 2-е изд. / Кузнецов С. Д. Интернет-Университет Информационных Технологий; Лаборатория знаний, 2007. — 484 с.
5. Васвани В. Zend Framework. Разработка веб-приложений на PHP / В. Васвани. Киев, 2012.–432 с.
6. Ланде Д.В. Новітні підходи й технології інформаційно-аналітичної підтримки прийняття рішень. // Національна безпека: український вимір: щокв. наук. зб. / Рада нац. безпеки і оборони України, Ін-т пробл. нац. безпеки; – К., 2008. – Вип. 1-2 (20-21). – С. 87-105.

Інтернет-ресурси

7. Мудрий Я. До проблеми створення web-сайту [Електронний ресурс]. – Режим доступу: http://eprints.zu.edu.ua/24816/1/Mudriy_APSI2017.pdf
8. [Електронний ресурс]. – Режим доступу: http://scbali.com/ua/web-studiya/typy-saytiv.html
9. Типи сайтів [Електронний ресурс]. – Режим доступу: https://www.greycampus.com/codelabs/php/OOPS

					КНТЕУ 121 06-05.БР			
Зм.	Аркуш	№ докум	Підпис	Дата				
Зав. кафедри		Криворучко О.В.			Розробка web-додатку управління логістичними послугами	Стадія	Аркуш	Аркушів
Керівник		Харченко О.А.				СД	43	43
Гарант		Цензура М. О.				Факультет інформаційних технологій, 4 курс, 6 група		
Розроб.		Євдокименко В.А.			Список використаних джерел			

ДОДАТКИ

```
<?php

namespace App\Repository;

use App\Entity\Car;
use Doctrine\Bundle\DoctrineBundle\Repository\ServiceEntityRepository;
use Doctrine\Persistence\ManagerRegistry;

/**
 * @method Car|null find($id, $lockMode = null, $lockVersion = null)
 * @method Car|null findOneBy(array $criteria, array $orderBy = null)
 * @method Car[]    findAll()
 * @method Car[]    findBy(array $criteria, array $orderBy = null, $limit = null, $offset = null)
 */
class CarRepository extends ServiceEntityRepository
{
    public function __construct(ManagerRegistry $registry)
    {
        parent::__construct($registry, Car::class);
    }


    public function getListWithPagination(int $limit = 1, int $offset = 0): array
    {
        return $this->createQueryBuilder('c')
```

```
            ->setMaxResults($limit)
            ->setFirstResult($offset)
            ->getQuery()
            ->getResult()
            ;
    }
}


<?php

namespace App\Repository;

use App\Entity\Country;
use Doctrine\Bundle\DoctrineBundle\Repository\ServiceEntityRepository;
use Doctrine\Persistence\ManagerRegistry;

/**
 * @method Country|null find($id, $lockMode = null, $lockVersion = null)
 * @method Country|null findOneBy(array $criteria, array $orderBy = null)
 * @method Country[]    findAll()
 * @method Country[]    findBy(array $criteria, array $orderBy = null, $limit = null, $offset = null)
 */
class CountryRepository extends ServiceEntityRepository
{
    public function __construct(ManagerRegistry $registry)
    {
        parent::__construct($registry, Country::class);
    }
}
```

```php
<?php

namespace App\Form;

use App\Entity\Car;
use Symfony\Component\Form\AbstractType;
use Symfony\Component\Form\FormBuilderInterface;
use Symfony\Component\OptionsResolver\OptionsResolver;
use Symfony\Component\Form\Extension\Core\Type\FileType;
use Symfony\Component\Validator\Constraints\File;
use Vich\UploaderBundle\Form\Type\VichImageType;

class CarType extends AbstractType
{
    public function buildForm(FormBuilderInterface $builder, array $options)
    {
        $builder
            ->add('imageFile', VichImageType::class, [
                'label' => 'image',

                // unmapped means that this field is not associated to any entity property
                'mapped' => true,

                // make it optional so you don't have to re-upload the PDF file
                // every time you edit the Product details
                'required' => false,
```

```
            // unmapped fields can't define their validation using annotations
            // in the associated entity, so you can use the PHP constraint classes
            'constraints' => [
                new File([
                    'maxSize' => '1024k',
                    'mimeTypes' => [
                        'image/jpeg',
                        'image/png',
                    ],
                    'mimeTypesMessage' => 'Please upload a valid PDF document',
                ])
            ],
        ])
        ->add('make')
        ->add('model')
        ->add('engine')
        ->add('fuel')
        ->add('transmission')
        ->add('year')
    ;
}

public function configureOptions(OptionsResolver $resolver)
{
    $resolver->setDefaults([
        'data_class' => Car::class,
    ]);
}
}


<?php
```

```php
namespace App\Entity;

use App\Repository\CountryRepository;
use Doctrine\ORM\Mapping as ORM;

/**
 * @ORM\Entity(repositoryClass=CountryRepository::class)
 */
class Country
{
    /**
     * @ORM\Id()
     * @ORM\GeneratedValue()
     * @ORM\Column(type="integer")
     */
    private $id;

    /**
     * @ORM\Column(type="string", length=255, nullable=true)
     */
    private $name;

    /**
     * @ORM\Column(type="float", nullable=true)
     */
    private $shipping_price;

    public function getId(): ?int
```

```
    {
        return $this->id;
    }

    public function getName(): ?string
    {
        return $this->name;
    }

    public function setName(?string $name): self
    {
        $this->name = $name;

        return $this;
    }

    public function getShippingPrice(): ?float
    {
        return $this->shipping_price;
    }

    public function setShippingPrice(?float $shipping_price): self
    {
        $this->shipping_price = $shipping_price;

        return $this;
    }
}
```

```php
<?php

namespace App\Entity;

use App\Repository\CarRepository;
use Doctrine\ORM\Mapping as ORM;
use Symfony\Component\HttpFoundation\File\File;
use Symfony\Component\HttpFoundation\File\UploadedFile;
use Vich\UploaderBundle\Entity\File as EmbeddedFile;
use Vich\UploaderBundle\Mapping\Annotation as Vich;

/**
 * @ORM\Entity(repositoryClass=CarRepository::class)
 * @Vich\Uploadable
 */
class Car
{
    /**
     * @ORM\Id()
     * @ORM\GeneratedValue()
     * @ORM\Column(type="integer")
     */
    private $id;

    /**
     * @ORM\Column(type="string", length=255, nullable=true)
     */
    private $make;
```

```
    /**
     * @ORM\Column(type="string", length=255, nullable=true)
     */
    private $model;


    /**
     * @ORM\Column(type="float", nullable=true)
     */
    private $engine;


    /**
     * @ORM\Column(type="string", length=255, nullable=true)
     */
    private $fuel;


    /**
     * @ORM\Column(type="string", length=255, nullable=true)
     */
    private $transmission;


    /**
     * NOTE: This is not a mapped field of entity metadata, just a simple property.
     *
     * @Vich\UploadableField(mapping="car_image", fileNameProperty="image.name", size="image.size", mimeType="image.mimeType", originalName="image.originalName", dimensions="image.dimensions")
     *
     * @var File|null
     */
```

```
    private $imageFile;


    /**
     * @ORM\Embedded(class="Vich\UploaderBundle\Entity\File")
     *
     * @var EmbeddedFile
     */
    private $image;


    /**
     * @ORM\Column(type="integer", nullable=true)
     */
    private $year;


    /**
     * @ORM\Column(type="float", nullable=true)
     */
    private $price;


    public function __construct()
    {
        $this->image = new EmbeddedFile();
    }


    /**
     * If manually uploading a file (i.e. not using Symfony Form) ensure an instance
     * of 'UploadedFile' is injected into this setter to trigger the  update. If this
     * bundle's configuration parameter 'inject_on_load' is set to 'true' this setter
     * must be able to accept an instance of 'File' as the bundle will inject one here
```

```
     * during Doctrine hydration.
     *
     * @param File|UploadedFile|null $imageFile
     */
    public function setImageFile(?File $imageFile = null)
    {
        $this->imageFile = $imageFile;
//
//        if (null !== $imageFile) {
//            // It is required that at least one field changes if you are using doctrine
//            // otherwise the event listeners won't be called and the file is lost
//            $this->updatedAt = new \DateTimeImmutable();
//        }
    }

    public function setImage(EmbeddedFile $image): void
    {
        $this->image = $image;
    }

    public function getImage(): ?EmbeddedFile
    {
        return $this->image;
    }

    public function getId(): ?int
    {
        return $this->id;
    }
```

```
public function getMake(): ?string
{
    return $this->make;
}

public function setMake(?string $make): self
{
    $this->make = $make;

    return $this;
}

public function getModel(): ?string
{
    return $this->model;
}

public function setModel(string $model): self
{
    $this->model = $model;

    return $this;
}

public function getEngine(): ?float
{
    return $this->engine;
}
```

```
public function setEngine(?float $engine): self
{
    $this->engine = $engine;

    return $this;
}

public function getFuel(): ?string
{
    return $this->fuel;
}

public function setFuel(?string $fuel): self
{
    $this->fuel = $fuel;

    return $this;
}

public function getTransmission(): ?string
{
    return $this->transmission;
}

public function setTransmission(?string $transmission): self
{
    $this->transmission = $transmission;
```

```
        return $this;
    }

    public function getYear(): ?int
    {
        return $this->year;
    }

    public function setYear(?int $year): self
    {
        $this->year = $year;

        return $this;
    }

    /**
     * @return File|null
     */
    public function getImageFile(): ?File
    {
        return $this->imageFile;
    }

    public function getPrice(): ?float
    {
        return $this->price;
    }
```

```
    public function setPrice(?float $price): self
    {
        $this->price = $price;

        return $this;
    }
}

<?php

namespace App\DataFixtures;

use App\Entity\Car;
use App\Entity\Country;
use Doctrine\Bundle\FixturesBundle\Fixture;
use Doctrine\Common\Persistence\ObjectManager;
use Symfony\Component\HttpFoundation\File\UploadedFile;

class CountryDataFixtures extends Fixture
{
    private const DATA_SET =
        [
            'USA' => 3000,
            'Canada' => 2700,
            'Poland' => 300,
            'Germany' => 500,
            'Japan' => 3700,
            'Korea' => 4000,
```

```
        'France' => 550,
        'Italy' => 550,
    ];

    public function load(ObjectManager $manager)
    {
        foreach (self::DATA_SET as $key => $value) {
            $manager->persist(
                (new Country())->setName($key)->setShippingPrice($value)
            );
        }

        $manager->flush();
    }
}


<?php

namespace App\DataFixtures;

use App\Entity\Car;
use Doctrine\Bundle\FixturesBundle\Fixture;
use Doctrine\Common\Persistence\ObjectManager;
use Symfony\Component\HttpFoundation\File\UploadedFile;
use function sprintf;

class CarDataFixtures extends Fixture
{
    private const DATA_SET =
```

```
[
    0 => [
        'make' => 'Toyota',
        'model' => 'Camry',
        'engine' => 3.5,
        'fuel' => 'бензин',
        'transmission' => 'автоматическая',
        'year' => 2019,
        'price' => 24000,
        'image' => 'camry19.jpg',
    ],
    1 => [
        'make' => 'Toyota',
        'model' => 'Corolla',
        'engine' => 2.0,
        'fuel' => 'бензин',
        'transmission' => 'автоматическая',
        'year' => 2013,
        'price' => 16000,
        'image' => 'corolla13.jpg',
    ],
    2 => [
        'make' => 'Toyota',
        'model' => 'Corolla',
        'engine' => 2.5,
        'fuel' => 'бензин',
        'transmission' => 'автоматическая',
        'year' => 2016,
        'price' => 22000,
        'image' => 'corolla16.jpg',
    ],
    3 => [
        'make' => 'Ford',
```

```
    'model' => 'Focus',
    'engine' => 1.0,
    'fuel' => 'бензин',
    'transmission' => 'автоматическая',
    'year' => 2018,
    'price' => 14000,
    'image' => 'focus18.jpg',
  ],
  4 => [
    'make' => 'Ford',
    'model' => 'Focus',
    'engine' => 2.5,
    'fuel' => 'бензин',
    'transmission' => 'автоматическая',
    'year' => 2019,
    'price' => 21000,
    'image' => 'focus19.jpg',
  ],
  5 => [
    'make' => 'Ford',
    'model' => 'Fusion',
    'engine' => 2.0,
    'fuel' => 'бензин',
    'transmission' => 'автоматическая',
    'year' => 2017,
    'price' => 23000,
    'image' => 'fusion17.jpg',
  ],
  6 => [
    'make' => 'Volkswagen',
    'model' => 'Golf VII',
    'engine' => 2.0,
    'fuel' => 'бензин',
```

```
    'transmission' => 'автоматическая',
    'year' => 2018,
    'price' => 15000,
    'image' => 'golfVII18.jpg',
  ],
  7 => [
    'make' => 'Hyundai',
    'model' => 'Elantra',
    'engine' => 2.0,
    'fuel' => 'бензин',
    'transmission' => 'автоматическая',
    'year' => 2018,
    'price' => 19000,
    'image' => 'hyundai-elantra18.jpg',
  ],
  8 => [
    'make' => 'Mercedes-Benz',
    'model' => 'cls',
    'engine' => 2.5,
    'fuel' => 'бензин',
    'transmission' => 'автоматическая',
    'year' => 2017,
    'price' => 30000,
    'image' => 'mersedes-cls17.jpg',
  ],
  9 => [
    'make' => 'Mercedes-Benz',
    'model' => 'cls',
    'engine' => 3.5,
    'fuel' => 'бензин',
    'transmission' => 'автоматическая',
    'year' => 2018,
    'price' => 38000,
```

```
        'image' => 'mersedes-cls18.jpg',
    ],
    10 => [
        'make' => 'Mercedes-Benz',
        'model' => 'cls',
        'engine' => 3.0,
        'fuel' => 'бензин',
        'transmission' => 'автоматическая',
        'year' => 2020,
        'price' => 60000,
        'image' => 'mersedes-cls20.jpg',
    ],
    11 => [
        'make' => 'Mercedes-Benz',
        'model' => 'gle',
        'engine' => 3.0,
        'fuel' => 'бензин',
        'transmission' => 'автоматическая',
        'year' => 2020,
        'price' => 80000,
        'image' => 'mersedes-gle20.jpg',
    ],
    12 => [
        'make' => 'Mercedes-Benz',
        'model' => 'gle-coupe',
        'engine' => 5.0,
        'fuel' => 'бензин',
        'transmission' => 'автоматическая',
        'year' => 2019,
        'price' => 92000,
        'image' => 'mersedes-gle-coupe19.jpg',
    ],
];
```

```
    public function load(ObjectManager $manager)
    {
        foreach (self::DATA_SET as $item) {
            $car = (new Car())
                ->setMake($item['make'])
                ->setModel($item['model'])
                ->setEngine($item['engine'])
                ->setFuel($item['fuel'])
                ->setTransmission($item['transmission'])
                ->setYear($item['year'])
                ->setPrice($item['price'])
            ;


            $car->setImageFile(
                new UploadedFile(
                    sprintf('%s/../../public/files/images/%s',__DIR__, $item['image']),
                    $item['image'],
                    null,
                    null,
                    true
                )
            );
            $manager->persist(
                $car
            );
        }


        $manager->flush();
    }
}
```

```php
<?php

namespace App\Controller;

use App\Repository\CountryRepository;
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;

class DeliveryCostController extends AbstractController
{
    /**
     * @Route("/delivery", name="delivery-cost")
     * @return Response
     */
    public function index(CountryRepository $countryRepository): Response
    {
        $countries = $countryRepository->findAll();


        return $this->render('delivery_cost.html.twig', ['countries'=> $countries]);
    }
}
<?php

namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
```

```
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;

class DefaultController extends AbstractController
{
    /**
     * @Route("/", name="index")
     * @return Response
     */
    public function index(): Response
    {
        return $this->render('index.html.twig');
    }
}
<?php

namespace App\Controller;

use App\Entity\Car;
use App\Form\CarType;
use App\Repository\CarRepository;
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;

/**
 * @Route("/car")
 */
class CarController extends AbstractController
```

```
{
    private const LIMIT_PER_PAGE = 6;

    /**
     * @Route("/", name="car_index", methods={"GET"})
     */
    public function index(CarRepository $carRepository, Request $request): Response
    {
        $offset = $request->query->getInt('offset', 0);
        $cars = $carRepository->getListWithPagination(self::LIMIT_PER_PAGE, $offset);
        $total = $carRepository->count([]);

        return $this->render(
            'car/index.html.twig',
            [
                'cars' => $cars,
                'total' => $total,
                'offset' => $offset,
                'limit' => self::LIMIT_PER_PAGE,
            ]
        );
    }

    /**
     * @Route("/new", name="car_new", methods={"GET","POST"})
     */
    public function new(Request $request): Response
    {
        $car = new Car();
        $form = $this->createForm(CarType::class, $car);
        $form->handleRequest($request);
```

```
    if ($form->isSubmitted() && $form->isValid()) {
        $entityManager = $this->getDoctrine()->getManager();
        $entityManager->persist($car);
        $entityManager->flush();

        return $this->redirectToRoute('car_index');
    }

    return $this->render(
        'car/new.html.twig',
        [
            'car' => $car,
            'form' => $form->createView(),
        ]
    );
}

/**
 * @Route("/{id}", name="car_show", methods={"GET"})
 */
public function show(Car $car): Response
{
    return $this->render(
        'car/show.html.twig',
        [
            'car' => $car,
        ]
    );
}
```

```
/**
 * @Route("/{id}/edit", name="car_edit", methods={"GET","POST"})
 */
public function edit(Request $request, Car $car): Response
{
    $form = $this->createForm(CarType::class, $car);
    $form->handleRequest($request);

    if ($form->isSubmitted() && $form->isValid()) {
        $this->getDoctrine()->getManager()->flush();

        return $this->redirectToRoute('car_index');
    }

    return $this->render(
        'car/edit.html.twig',
        [
            'car' => $car,
            'form' => $form->createView(),
        ]
    );
}

/**
 * @Route("/{id}", name="car_delete", methods={"DELETE"})
 */
public function delete(Request $request, Car $car): Response
{
```

```
        if ($this->isCsrfTokenValid('delete' . $car->getId(), $request->request->get('_token'))) {
            $entityManager = $this->getDoctrine()->getManager();
            $entityManager->remove($car);
            $entityManager->flush();
        }

        return $this->redirectToRoute('car_index');
    }
}
```

```
{% extends 'base.html.twig' %}

{% block body %}

    <div class="container">

        <div class="row" style="height: 30vh;">
            <p style="font-size: 13pt; text-align: center;  padding-top: 50px ">
                Здесь вы можете поссчитать итоговую стоимость автомобиля. </br>Внимание! Это приблизительная стоимость, за точной суммой обратитесь к менеджеру.
                Для просчета условной итоговой стоимости заполните давнные.
            </p>
        </div>
        <div style=" background: url('/files/images/logotypes/plane-ship-car.jpg') no-repeat center/cover; min-height: 500px; padding: 100px 0 100px 0">
            <div class="row" style="text-align: center; margin: 30px 0 0 0 ">
                <div class="col-6">Объем двигателя в куб. см.</div>
                <div class="col-6">
                    <div style="width: 150px">
                        <input type="text" class="form-control" id="engine" aria-describedby="emailHelp" placeholder="куб.см.">
```

```
            </div>
          </div>
        </div>
        <div class="row" style="text-align: center; margin: 30px 0 0 0 ">
          <div class="col-6">Кол-во лет автомобилю</div>
          <div class="col-6">
            <div style="width: 150px">
              <input type="text" class="form-control" id="years" aria-describedby="emailHelp"
placeholder="кол-во лет">
            </div>
          </div>
        </div>
        <div class="row" style="text-align: center; margin: 30px 0 0 0 ">
          <div class="col-6">Вид топлива</div>
          <div class="col-6">
            <div style="width: 150px">
              <select class="form-control" id="fuel">
                <option value="petrol" selected>Бензин</option>
                <option value="diesel">Дизель</option>
              </select>
            </div>
          </div>
        </div>
        <div class="row" style="text-align: center; margin: 30px 0 0 0 ">
          <div class="col-6">Выберите страну</div>
          <div class="col-6">
            <div style="width: 150px">
              <select class="form-control" id="country">
                {% for country in countries %}
                  <option value='{{ country.shippingPrice }}'>{{ country.name }}</option>
                {% endfor %}
              </select>
```

```
                </div>
            </div>
        </div>
        <div class="row" style="text-align: center; margin: 30px 0 0 0 ">
            <div class="col-6">Укажите начальную стоимость авто.</div>
            <div class="col-6">
                <div style="width: 150px">
                    <input type="text" class="form-control" id="startPrice" aria-describedby="emailHelp" placeholder="$">
                </div>
            </div>
        </div>
        <div class="row" style="text-align: center; margin: 30px 0 0 0 ">
            <div class="col-3"></div>
            <div class="col-3">
                <div style="width: 150px">
                    <button type="button" id="calculate" class="btn btn-dark">Посчитать</button>
                </div>
            </div>
            <div class="col-6"></div>
        </div>
    </div>
    <div style="height: 50px" id="responseAmount"></div>
</div>
{% endblock %}
{% block javascripts %}


<script type="text/javascript">


    $(function () {
        $('#calculate').on('click', function (e) {
            let engineSize = parseFloat($('#engine').val());
```

```
        if (!engineSize || isNaN(engineSize)) {
            alert("Укажите правильный объем двигателя");
            return;
        }
        let yearsCount = parseInt($('#years').val());
        if (!yearsCount || isNaN(engineSize)) {
            alert("Укажите сколько лет автомобилю");
            return;
        } else if (yearsCount === 0) {
            yearsCount = 1
        }
        let fuel = $("#fuel option:selected").val();
        if (!fuel) {
            alert("Укажите вид топлива");
            return;
        }
        let countryShippingPrice = parseInt($('#country').val());
        if (!countryShippingPrice || isNaN(countryShippingPrice)) {
            alert("Укажите страну");
            return;
        }
        let startPrice = parseFloat($('#startPrice').val());
        if (!startPrice || isNaN(startPrice)) {
            alert("Укажите правильный объем двигателя");
            return;
        }
        const amountWithoutCountryShipping = parseFloat(startPrice) +
parseFloat(getTaxAmount(engineSize, yearsCount, fuel, startPrice));
        const totalAmount = parseFloat(countryShippingPrice) +
parseFloat(amountWithoutCountryShipping) + parseFloat(1000);


        const messageResultAmount =
```

```
            'Учитывая НДС, пошлинную и таможенные сборы Ваш автомобиль будет стоить '
            + amountWithoutCountryShipping
            + '. С доставкой и услугами компании '
            + totalAmount
        ;


        $('#responseAmount').text(messageResultAmount);
    });


});


function getTaxAmount(engineSize, countYears, fuelType, startPrice) {
    const ndsPercent = 20;
    const tollPercent = 10;            //poshlinaya
    const retirementFeePercent = 5;    //pensionniy

    const axcisTax = getAxcisTaxAmount(engineSize, countYears, fuelType, startPrice,);
    const tollTax = startPrice / 100 * tollPercent;
    const ndsTax = (startPrice + axcisTax + tollTax) / 100 * ndsPercent;
    const retirementFeeTax = startPrice / 100 * retirementFeePercent;

    return parseFloat(axcisTax) + parseFloat(tollTax) + parseFloat(ndsTax) + parseFloat(retirementFeeTax);
}


function getAxcisTaxAmount(engineSize, countYears, fuelType) {
    let engineTax = 0;
    if (fuelType == 'petrol') {
```

```
            engineTax = engineSize < 3000 ? 50 : 100;
        } else if (fuelType == 'diesel') {
            engineTax = engineSize < 3000 ? 75 : 150;
        }
        const engineCoefficient = engineSize / 1000;
        const returnCoefficient = countYears;


        return engineTax * engineCoefficient * returnCoefficient;
    }
</script>
{% endblock %}
```

					КНТЕУ 121 06-05.БР			
Зм.	Аркуш	№ докум	Підпис	Дата				
Зав. кафедри		Криворучко О.В.			Розробка web-додатку управління логістичними послугами	Стадія	Аркуш	Аркушів
Керівник		Харченко О.А.				СД	75	43
Гарант		Цензура М. О.				Факультет інформаційних технологій, 4 курс, 6 група		
Розроб.		Євдокименко В.А.			Список використаних джерел			

Київський національний торговельно-економічний університет

Факультет *інформаційних технологій*
Кафедра *інженерії програмного забезпечення та кібербезпеки*
Освітній ступінь *бакалавр*
Спеціальність *121 «Інженерія програмного забезпечення»*
Спеціалізація / освітня програма *121 «Інженерія програмного забезпечення»*

Затверджую
Зав. кафедри інженерії
програмного забезпечення
та кібербезпеки
Криворучко О. В.
"7" листопада 2019 р.

Завдання
на випускну кваліфікаційну роботу студентові

Євдокименка Володимира Андрійовича

1. Тема випускної кваліфікаційної роботи *Розробка web-додатку управління логістичними послугами*
Затверджена наказом ректора від *«02» грудня 2019 р. № 4112*

2. Строк здачі студентом закінченої роботи _____
_____

3. Цільова установка та вихідні дані до роботи

Об'єктом дослідження *є діяльність логістичних компаній*

Предметом дослідження *є методи та алгоритми web-додатку*

Мета роботи *дослідження можливостей у сфері web-розробки, набуття навичок щодо застосування мов web-програмування, середовищ розробки та спеціалізованих текстових редакторів, систем управління базами даних тощо.*

4. Консультанти роботи із зазначенням розділів, які консультують:

Розділ	Консультант (прізвище, ініціали)	Підпис, дата	
		Завдання видав	Завдання прийняв

5. Зміст випускної кваліфікаційної роботи (перелік питань за кожним розділом)

6. Календарний план виконання роботи

№ пор.	Назва етапів випускної кваліфікаційної роботи	Строк виконання етапів роботи	
		за планом	фактично
1	2	3	4
1.	*Вибір теми випускної кваліфікаційної роботи*		
2.	*Вступ та перелік літературних джерел*		
3.	*Розділ 1. «З чого складається сайт, кому він потрібень та як його створенний»*		
4.	*Розділ 2. «Особливості розробки web-додатку»*		
5.	*Розділ 3. «Розробка сайту»*		
6.	*Висновки*		
7.	*Здача випускної кваліфікаційної роботи на кафедру (перша перевірка)*		
8.	*Підготовка автореферату та презентації доповіді*		
9.	*Попередній захист випускної кваліфікаційної роботи*		
10.	*Зовнішнє рецензування випускної кваліфікаційної роботи*		
11.	*Здача прошитої випускної кваліфікаційної роботи на кафедру*		
12.	*Публічний захист випускної кваліфікаційної роботи*		

7. Дата видачі завдання «___»_____20__ р.

8. Науковий керівник випускної кваліфікаційної роботи Харченко О. А.

9. Гарант освітньої програми Цензура М. О.

10. Завдання прийняв до виконання студент Євдокименко В. А.

11. Відгук керівника випускної кваліфікаційної роботи

Науковий керівник випускної кваліфікаційної роботи _____
(підпис, дата)

Відмітка про попередній захист *Харченко О. А.* _____
(ПІБ, підпис, дата)

12. Висновок про випускну кваліфікаційну робота

Випускна кваліфікаційна робота студентки *Євдокименко В. А.* може бути допущена до захисту екзаменаційній комісії.

Гарант освітньої програми Цензура М. О.

Завідувач кафедри Криворучко О. В.

«_____»_____20 ___ р.