Молдавский Государственный Университет

Факультет Математики и Информатики

Прикладная Информатика

**Индивидуальная работа по Symfony**

Framework-uri pentru Dezvoltarea Aplicațiilor Web

На тему: **“Прокат автомобилей”**

Выполнила: Лесная Екатерина

Группа: IA2003

Проверил преподаватель: Светлана Бодруг

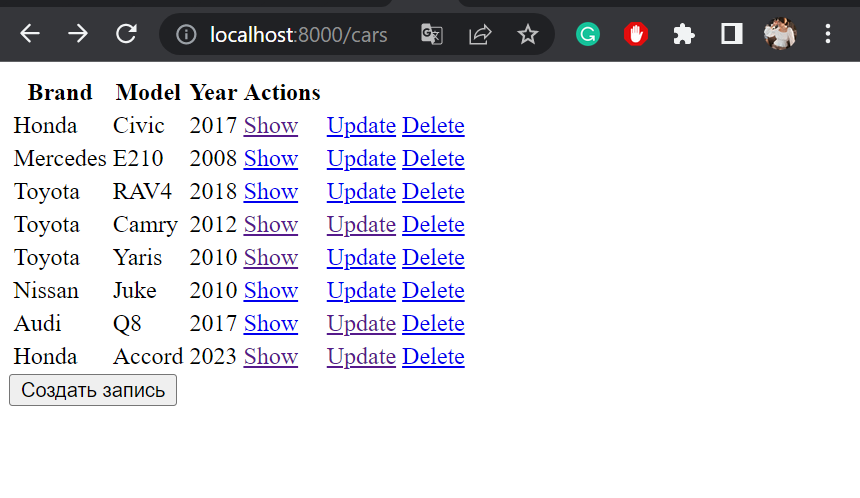
Кишинев 2022

Прокат автомобилей

Для выполнения данной лабораторной работы были заранее предусмотренные необходимые требования для реализации, а их стилизация и расположение на веб сайте оставались на усмотрение студента.

# 1. Главная страница

На главной странице располагается меню нашего приложения



**Фрагмент кода программы отображения нашей формы**

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

{% block title%} Cars {% endblock %}

{% block body %}

{% if cars %}

<table class="table table-striped">

<thead>

<tr>

<th>Brand</th>

<th>Model</th>

<th>Year</th>

<th>Actions</th>

</tr>

</thead>

<tbody>

{% for car in cars %}

<tr>

<td>{{ car.brandId.name }}</td>

<td>{{ car.model }}</td>

<td>{{ car.year }}</td>

<td>

<a href="/car/{{ car.id }}" class="btn btn-dark">Show</a>

</td>

<td>

<a href="/car/update/{{ car.id }}" class="btn btn-info">Update</a>

</td>

<td>

<a href="/car/delete/{{ car.id }}" class="btn btn-danger">Delete</a>

</td>

</tr>

{% endfor %}

</tbody>

</table>

<form action="/car">

<input type="submit" value="Создать запись" />

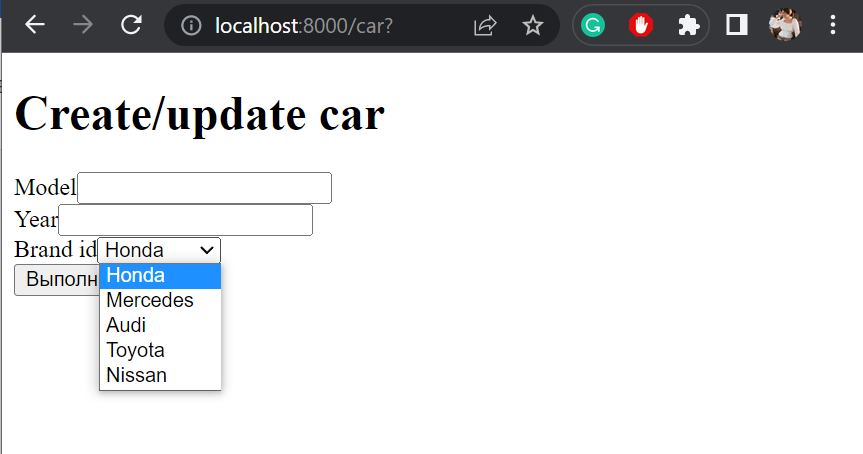
</form>

{% else %}

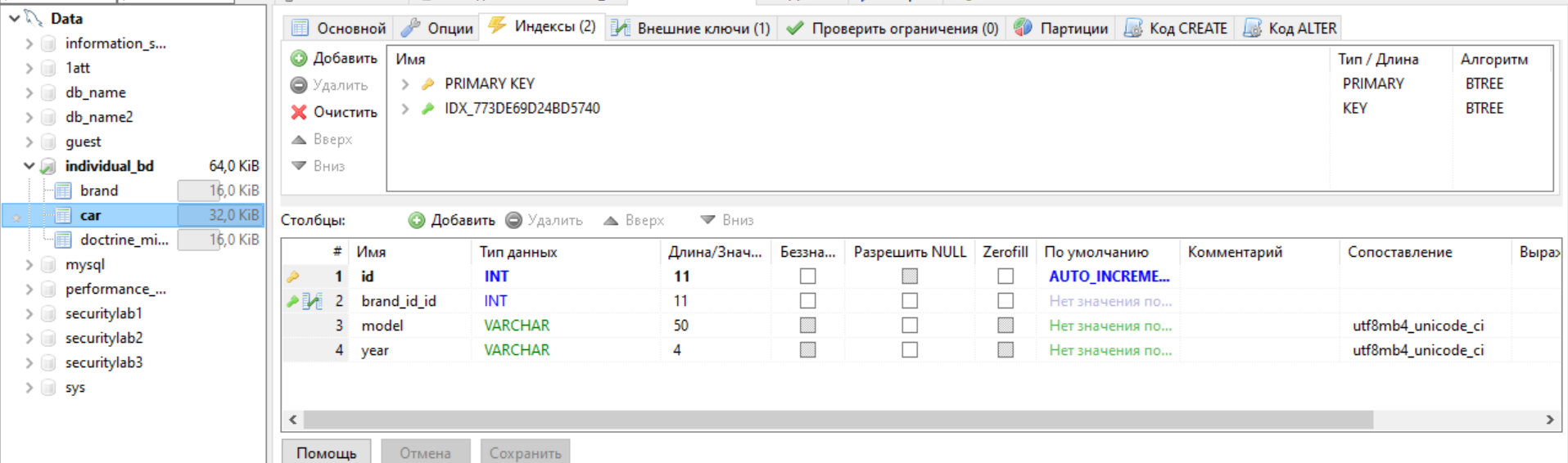
{% endif %}

{% endblock %}

# 2. Страница добавления/модификации данных



# 3. База Данных



**Общий фрагмент кода программы (CRUD):**

<?php

namespace App\Controller;

use App\Entity\Car;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;

use Symfony\Component\Form\Extension\Core\Type\SubmitType;

use Symfony\Component\Form\Extension\Core\Type\TextType;

use Symfony\Component\HttpFoundation\Request;

use Symfony\Component\HttpFoundation\Response;

use Symfony\Component\Routing\Annotation\Route;

use Sensio\Bundle\FrameworkExtraBundle\Configuration\Method;

use Doctrine\Persistence\ManagerRegistry;

use App\Service\CarService;

use Symfony\Component\Form\Extension\Core\Type\FormType;

use App\Form\CarFormType;

use Psr\Log\LoggerInterface;

class CarController extends AbstractController

{

    #[Route('/')]

    public function index(LoggerInterface $logger): Response

    {

        $logger->info("Index page: user IP " . $\_SERVER['REMOTE\_ADDR']);

        return $this->render('car/index.html.twig', []);

    }

    #[Route('/cars', name: "display\_cars")]

    public function displayCars(ManagerRegistry $doctrine, LoggerInterface $logger):Response

    {

        $car = new CarService($doctrine->getManager(), Car::class);

        $cars = $car->getAllCars();

        $logger->info("List of cars page: user IP " . $\_SERVER['REMOTE\_ADDR']);

        return $this->render('default/cars.html.twig', array

        ('cars' => $cars));

    }

    #[Route('/car', name: 'create\_car')]

    public function createCar(Request $request, ManagerRegistry $doctrine, LoggerInterface $logger):Response

    {

        $car = new Car();

        $form = $this->createForm(CarFormType::class, $car);

        $form->handleRequest($request);

        if($form->isSubmitted() && $form->isValid()){

            $car = $form->getData();

            $entityManager = $doctrine->getManager();

            $entityManager->persist($car);

            $entityManager->flush();

            $logger->info("Create car action: user IP " . $\_SERVER['REMOTE\_ADDR']);

            return $this->redirectToRoute('display\_cars');

        }

        $logger->info("Create car action: user IP " . $\_SERVER['REMOTE\_ADDR']);

        return $this->render('default/form.html.twig',array(

            'form'=>$form->createView()

        ));

    }

    #[Route('/car/{id}', name: 'display\_car')]

    public function displayCar(int $id, ManagerRegistry $doctrine, LoggerInterface $logger):Response

    {

        $car  = new CarService($doctrine->getManager(),Car::class);

        $car = $car->getCar($id);

        $logger->info("Car page: user IP " . $\_SERVER['REMOTE\_ADDR']);

        return $this->render('car/display.html.twig', array('car' =>$car));

    }

    #[Route('/car/update/{id}', name: 'update\_car')]

    public function updateCar( Request $request, $id, ManagerRegistry $doctrine, LoggerInterface $logger):Response

    {

        $car = $doctrine->getRepository(Car::class)->find($id);

        $form = $this->createForm(CarFormType::class, $car);

        $form->handleRequest($request);

        if($form->isSubmitted() && $form->isValid()){

            $carService = new CarService($doctrine->getManager(), Car::class);

            $carService->addCar($car);

            $logger->info("Update car action: user IP " . $\_SERVER['REMOTE\_ADDR']);

            return $this->redirectToRoute('display\_cars');

        }

        $logger->info("Update car action: user IP " . $\_SERVER['REMOTE\_ADDR']);

        return $this->render('default/form.html.twig',array(

            'form'=>$form->createView()

        ));

    }

    #[Route('/car/delete/{id}', name: 'delete\_car')]

    public function deleteCar(Request $request, $id, ManagerRegistry $doctrine, LoggerInterface $logger):Response

    {

        $car = new CarService($doctrine->getManager(), Car::class);

        $car->deleteCar($id);

        $logger->info("Delete car action: user IP " . $\_SERVER['REMOTE\_ADDR']);

        return $this->redirectToRoute('display\_cars');

    }

}

# 4. Конфигурация нашего проекта

composer create-project symfony/skeleton lesnaia

(не забываем перейти в созданную папку)

composer require twig

composer require annotations

composer require profiler (отладка)

composer require logger (система логгирования)

composer require monolog/monolog (использование логгера в контроллере через LoggerInterface)

!!!!composer require doctrine(не делали)

composer require symfony/form

Базы данных

composer require symfony/orm-pack

composer require --dev symfony/maker-bundle

======================================================================

php bin/console make:entity

Class name of the entity to create or update (e.g. OrangeKangaroo):

> Brand

created: src/Entity/Brand.php

created: src/Repository/BrandRepository.php

Entity generated! Now let's add some fields!

You can always add more fields later manually or by re-running this command.

New property name (press <return> to stop adding fields):

> name

Field type (enter ? to see all types) [string]:

>

Field length [255]:

> 50

Can this field be null in the database (nullable) (yes/no) [no]:

>

updated: src/Entity/Brand.php

Add another property? Enter the property name (or press <return> to stop adding fields):

>

===========================================================================

php bin/console make:entity

Class name of the entity to create or update (e.g. GentlePopsicle):

> Car

created: src/Entity/Car.php

created: src/Repository/CarRepository.php

Entity generated! Now let's add some fields!

You can always add more fields later manually or by re-running this command.

New property name (press <return> to stop adding fields):

> model

Field type (enter ? to see all types) [string]:

>

Field length [255]:

> 50

Can this field be null in the database (nullable) (yes/no) [no]:

>

updated: src/Entity/Car.php

Add another property? Enter the property name (or press <return> to stop adding fields):

> year

Field type (enter ? to see all types) [string]:

>

Field length [255]:

> 4

Can this field be null in the database (nullable) (yes/no) [no]:

>

updated: src/Entity/Car.php

Add another property? Enter the property name (or press <return> to stop adding fields):

> brand\_id

Field type (enter ? to see all types) [integer]:

> relation

What class should this entity be related to?:

> Brand

What type of relationship is this?

------------ -----------------------------------------------------------------

Type Description

------------ -----------------------------------------------------------------

ManyToOne Each Car relates to (has) one Brand.

Each Brand can relate to (can have) many Car objects.

OneToMany Each Car can relate to (can have) many Brand objects.

Each Brand relates to (has) one Car.

ManyToMany Each Car can relate to (can have) many Brand objects.

Each Brand can also relate to (can also have) many Car objects.

OneToOne Each Car relates to (has) exactly one Brand.

Each Brand also relates to (has) exactly one Car.

------------ -----------------------------------------------------------------

Relation type? [ManyToOne, OneToMany, ManyToMany, OneToOne]:

> ManyToOne

Is the Car.brand\_id property allowed to be null (nullable)? (yes/no) [yes]:

> no

Do you want to add a new property to Brand so that you can access/update Car objects from it - e.g. $brand->getCars()? (yes/no) [yes]:

> yes

A new property will also be added to the Brand class so that you can access the related Car objects from it.

New field name inside Brand [cars]:

>

Do you want to activate orphanRemoval on your relationship?

A Car is "orphaned" when it is removed from its related Brand.

e.g. $brand->removeCar($car)

NOTE: If a Car may \*change\* from one Brand to another, answer "no".

Do you want to automatically delete orphaned App\Entity\Car objects (orphanRemoval)? (yes/no) [no]:

> no

updated: src/Entity/Car.php

updated: src/Entity/Brand.php

Add another property? Enter the property name (or press <return> to stop adding fields):

>

====================================================================================

php bin/console make:migration

Success!

php bin/console doctrine:migrations:migrate

WARNING! You are about to execute a migration in database "individual\_bd" that could result in schema changes and data loss.

Are you sure you wish to continue? (yes/no) [yes]:

>

====================================================================================

php bin/console make:controller CarController