Министерство образования Республики Беларусь
Учреждение образования
«Брестский государственный технический университет» Кафедра ИИТ

Отчет по лабораторной работе 6
Дисциплина “ПрИС”

Выполнил:

Студент группы ПО-3

Кабачук Д. С.

Проверил:

Лаврущик А. И.

Брест 2021

Вариант 11

Цель работы: Познакомиться с практической реализацией принципа инверсии зависимостей, принципа единственной ответственности и механизмом внедрения зависимостей.

Постановка задачи: Реализуйте сервисы на уровне приложения, реализующие с доменными моделями те же действия, что и сценарий транзакции из ЛР No1. Хотя бы одно действие должно реализовываться доменным сервисом.

Предметная область: космический туризм.

Ход работы

PassengersController.php:

```
<?php

namespace App\Controller;

use App\Entity\Passenger;
use App\Entity\Travel;
use App\Form\PassengerType;
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\Routing\Annotation\Route;

class PassengersController extends AbstractController
{
    /**
     * @Route("/create_passenger", name="create_passenger")
     */
    public function createPassenger(Request $request)
    {
        $em = $this->getDoctrine()->getManager();

        $passenger = new Passenger();
        $tixes = $em->getRepository(Travel::class)->findAll();
        $form = $this->createForm(PassengerType::class, $passenger);
        $form->handleRequest($request);

        if($form->isSubmitted() && $form->isValid()) {
            $data = $form->getData();
            $em->persist($passenger);
            $em->flush();
        }

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

    /**
```

```
     * @Route("/update_passenger/{id}", name="update_passenger")
     */
    public function updatePassenger(Request $request, $id)
    {
        $em = $this->getDoctrine()->getManager();

        $passenger =  $em->getRepository(Passenger::class)->find($id);
        $tixes = $em->getRepository(Travel::class)->findAll();
        $form = $this->createForm(PassengerType::class, $passenger);
        $form->handleRequest($request);

        if($form->isSubmitted() && $form->isValid()) {
            $data = $form->getData();
            $em->persist($passenger);
            $em->flush();
        }

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


    /**
     * @Route("/delete_passenger/{id}", name="delete_passenger")
     */
    public function deletePassenger($id)
    {
        $em = $this->getDoctrine()->getManager();
        $passenger = $em->getRepository(Passenger::class)->find($id);
        $em->remove($passenger);
        $em->flush();
        return $this->redirectToRoute('all_passengers');
    }

    /**
     * @Route("/all_passengers", name="all_passengers")
     */
    public function allPassengers()
    {
        $em = $this->getDoctrine()->getManager();
        $passengers = $em->getRepository(Passenger::class)->findAll();
        return $this->render('passenger/passenger_all.html.twig', [
            'passengers' => $passengers,
        ]);
    }
}
```

TravelsController.php:

```
<?php

namespace App\Controller;

use App\Domains\ChooseShip;
use App\Entity\Passenger;
```

```
use App\Entity\Ship;
use App\Entity\Travel;
use App\Form\TravelType;
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\Routing\Annotation\Route;

class TravelsController extends AbstractController
{

    /**
     * @Route("/create_travel", name="create_travel")
     */
    public function createTravel(Request $request)
    {
        $em = $this->getDoctrine()->getManager();

        $ships = $em->getRepository(Ship::class)->findAll();
        $travel = new Travel();
        $form = $this->createForm(TravelType::class, $travel);
        $form->handleRequest($request);

        if($form->isSubmitted() && $form->isValid()) {
            try {
                $data = $form->getData();
                $chooseShip = new ChooseShip($data->getShipId(), $em);
                $ship = $chooseShip->getShip();
                $travel->setShip($ship);
                $em->persist($ship);
                $em->persist($travel);
                $em->flush();
            }catch (\Error $exception) {
                return $this->render('travel/travel_form.html.twig', [
                    'form' => $form->createView(),
                    'ships' => $ships,
                    'message' => $exception->getMessage(),
                ]);
            }
        }

        return $this->render('travel/travel_form.html.twig', [
            'form' => $form->createView(),
            'ships' => $ships,
            'message' => "",
        ]);
    }

    /**
     * @Route("/update_travel/{id}", name="update_travel")
     */
    public function updateTravel(Request $request, $id)
    {
        $em = $this->getDoctrine()->getManager();

        $ships = $em->getRepository(Ship::class)->findAll();
        $travel = $em->getRepository(Travel::class)->find($id);
        $form = $this->createForm(TravelType::class, $travel);
        $form->handleRequest($request);
```

```
        if($form->isSubmitted() && $form->isValid()) {
            $data = $form->getData();
            $ship = $em->getRepository(Ship::class)->find($data->getShipId());
            $currentNum = $ship->getCurrentNumber();
            if ($currentNum === $ship->getCapacity()) {
                return $this->render('travel/travel_form.html.twig', [
                    'form' => $form->createView(),
                    'ships' => $ships,
                    'message' => "Ship is full!!!",
                ]);
            }
            $ship->setCurrentNumber($currentNum + 1);
            $travel->setShip($ship);
            $em->persist($ship);
            $em->persist($travel);
            $em->flush();
        }

        return $this->render('travel/travel_update_form.html.twig', [
            'form' => $form->createView(),
            'ships' => $ships,
            'message' => "",
        ]);
    }

    /**
     * @Route("/delete_travel/{id}", name="delete_travel")
     */
    public function deleteTravel($id)
    {
        $em = $this->getDoctrine()->getManager();
        $travel = $em->getRepository(Travel::class)->find($id);
        $ship = $em->getRepository(Ship::class)->find($travel->getShipId);
        $passengers = $em->getRepository(Passenger::class)->findBy(["tix" =>
$id]);
        foreach ($passengers as $passenger) {
            $em->remove($passenger);
        }
        $currentNum = $ship->getCurrentNumber();
        $ship->setCurrentNumber($currentNum + 1);
        $em->persist($ship);
        $em->remove($travel);
        $em->flush();
        return $this->redirectToRoute('all_travels');
    }

    /**
     * @Route("/all_travels", name="all_travels")
     */
    public function allTravels()
    {
        $em = $this->getDoctrine()->getManager();
        $travels = $em->getRepository(Travel::class)->findAll();
        return $this->render('travel/travel_all.html.twig', [
            'travels' => $travels,
        ]);
    }
```

```
}
```

ShipsController.php:

```
<?php

namespace App\Controller;

use App\Entity\Passenger;
use App\Entity\Ship;
use App\Entity\Travel;
use App\Form\ShipType;
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\Routing\Annotation\Route;

class ShipsController extends AbstractController
{
    /**
     * @Route("/create_ship", name="create_ship")
     */
    public function createShip(Request $request)
    {
        $em = $this->getDoctrine()->getManager();
        $ship = new Ship();
        $form = $this->createForm(ShipType::class, $ship);
        $form->handleRequest($request);

        if($form->isSubmitted() && $form->isValid()) {
            $data = $form->getData();
            $ship->setCurrentNumber(0);
            $em->persist($ship);
            $em->flush();

        }

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

    /**
     * @Route("/update_ship/{id}", name="update_ship")
     */
    public function updateShip(Request $request, $id)
    {
        $em = $this->getDoctrine()->getManager();
        $ship = $em->getRepository(Ship::class)->find($id);
        $form = $this->createForm(ShipType::class, $ship);
        $form->handleRequest($request);

        if($form->isSubmitted() && $form->isValid()) {
            $data = $form->getData();
            $ship->setCurrentNumber(0);
            $em->persist($ship);
            $em->flush();

        }
```

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

    /**
     * @Route("/delete_ship/{id}", name="delete_ships")
     */
    public function deleteShip($id)
    {
        $em = $this->getDoctrine()->getManager();
        $ship = $em->getRepository(Ship::class)->find($id);
        $travels = $em->getRepository(Travel::class)->findBy(['ship_id' =>
$id]);
        foreach ($travels as $travel) {
            $passengers = $em->getRepository(Passenger::class)->findBy(["tix" =>
$travel->getId()]);
            foreach ($passengers as $passenger) {
                $em->remove($passenger);
            }
            $em->remove($travel);
        }
        $em->remove($ship);
        $em->flush();
        return $this->redirectToRoute('all_ships');
    }

    /**
     * @Route("/all_ships", name="all_ships")
     */
    public function allShips()
    {
        $em = $this->getDoctrine()->getManager();
        $ships = $em->getRepository(Ship::class)->findAll();
        return $this->render('ship/ship_all.html.twig', [
            'ships' => $ships,
        ]);
    }
}
```

Вывод: Я познакомился с практической реализацией принципа инверсии зависимостей, принципа единственной ответственности и механизмом внедрения зависимостей.