Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования «Ульяновский государственный технический университет»

Кафедра «Вычислительная техника»

**Отчет по лабораторной работе №7**

Дисциплина: «Разработка профессиональных приложений»

Вариант №5

Выполнил:

студент группы ИВТАСбд-22

Зейнетдинов М.М.

Проверил:

преподаватель кафедры

«Вычислительная техника»

Исхаков И.И.

Ульяновск, 2023

Задание по варианту

Для предложенного варианта заданий необходимо реализовать веб-сайт в среде Django, удовлетворяющий следующим требованиям:

1. Количество таблиц в БД - не менее 5. Таблицы должны быть спроектированы в соответствии с тремя нормальными формами (https://habr.com/ru/post/254773/).

2. Должны быть реализованы 3 типа связей: one-to-one, one-to-many, man-to-many.

3. Таблицы создаются с использованием моделей в Django и миграций.

4. Как минимум, для 1 сущности должны быть реализованы CRUD операции (Create, read, update, delete) + представление списка сущностей.

5. Остальные таблицы должны быть доступны для редактирования в административной части сайта.

6. Необходимые пять таблиц должны быть разработаны для моделирования предметной области, заданной вариантом. Необходимо придумать эти таблицы, не забывая про здравый смысл. Например, для варианта 1 (Студенты: №, ФИО, email, группа) можно сделать следующие таблицы: 1 - группы студентов, 2 - студенты, 3 - направления подготовки, 4 - факультеты, 5 - изучаемые дисциплины.

Описание реализации

Была создана база данных, которая содержит информацию о чеке, товарах, продавцах, магазинах и продажах, а затем перенесена в файл models.py, где были прописаны таблицы и связи между ними. Для того, чтобы эти изменения были применены к базе данных, были использованы команды pyhton manage.py makemigrations и pyhton manage.py migrate.

Для взаимодействия с базой данных была использована библиотека Django, которая предоставляет удобные средства для работы с моделями данных. В рамках данной базы данных необходимо было реализовать операции CRUD (создание, чтение, обновление и удаление данных), которые были успешно выполнены с помощью функционала библиотеки Django.

Также были созданы URL-адреса (url) для каждой из сущностей, что позволяет пользователям переходить между страницами и взаимодействовать с базой данных. В целом, благодаря Django была создана удобная и функциональная система для работы с базой данных.

Для управления всей системой был создан superuser, который имеет права на редактирование данных и доступ к административной панели Django. Это позволяет управлять базой данных и ее содержимым в удобном интерфейсе.

**Описание возникших затруднений**

Затруднения возникли с изучением работы фреймворков Django. А именно как и где правильно прописать таблицу и связать их между собой, после чего понять как это всё превратить в бд и правильно отображать на сайте с возможностью редактирования.

Описание альтернативных способов решения

Можно было использовать другие таблицы, например вид магазина. Расширить поля в таблице, например почтовый адрес продавцов, возраст. Ну или же самому создать базу данных не используя миграции, например на другой СУБД.

Тестирование

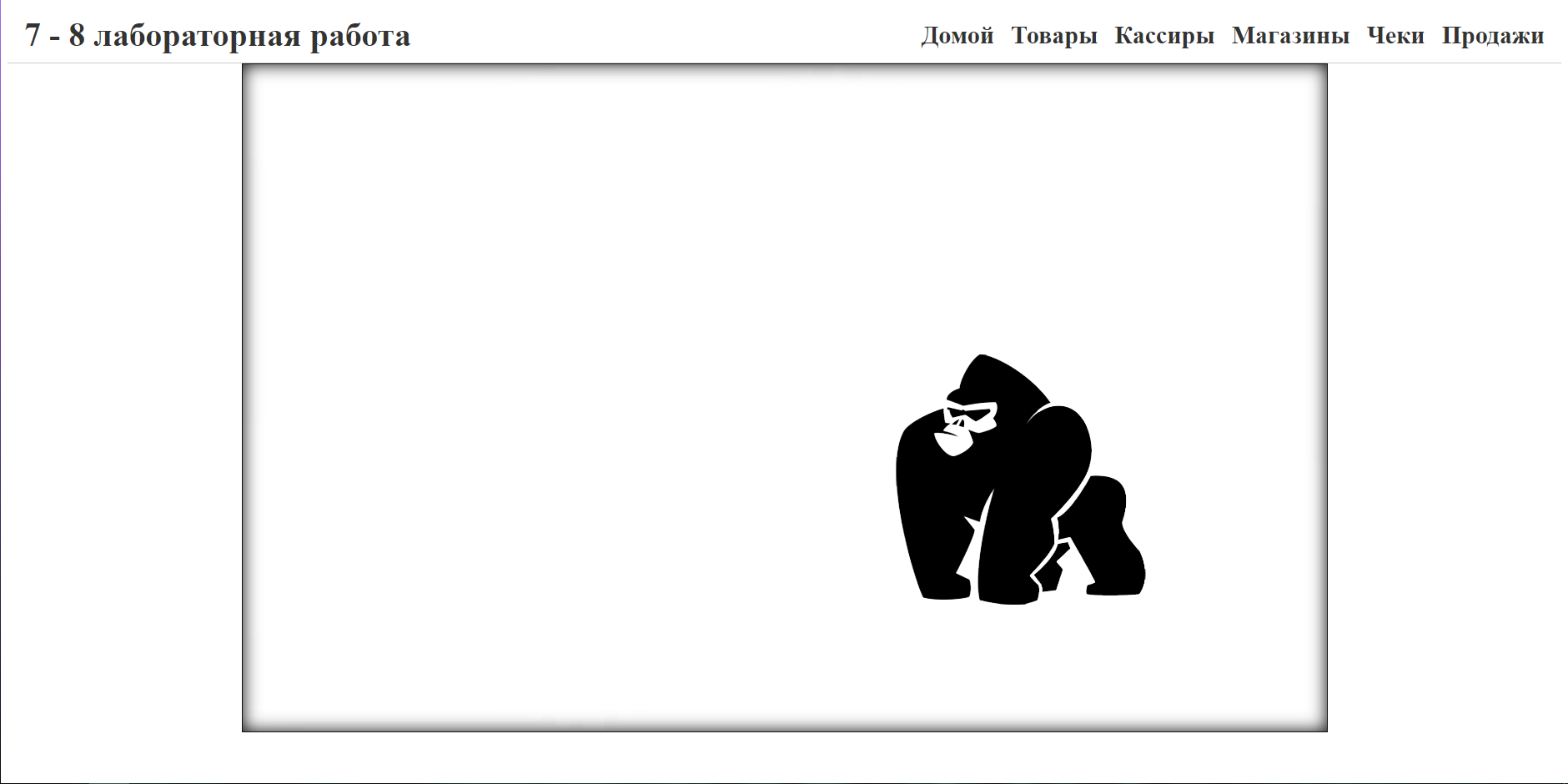
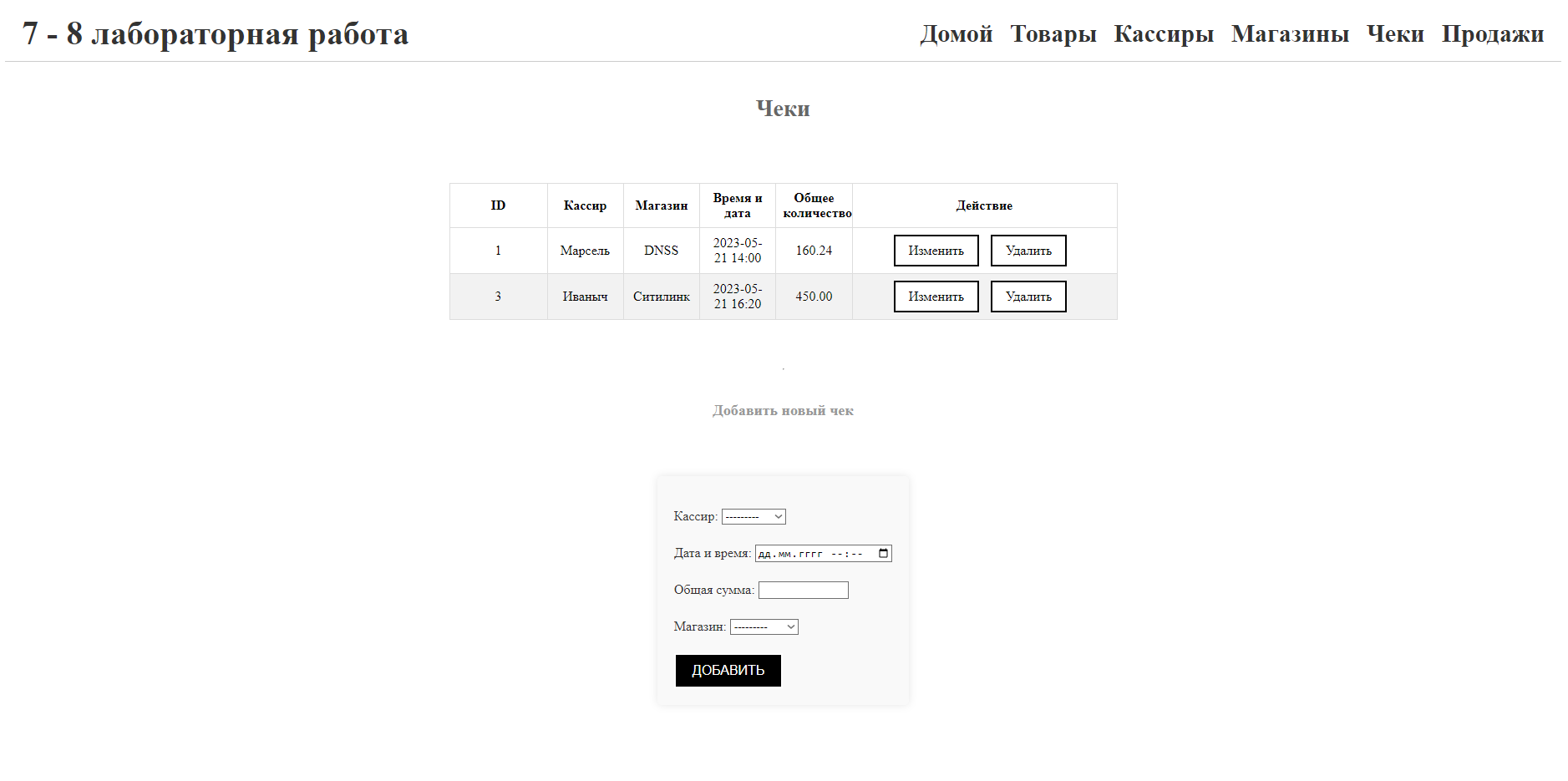


Рис. 1 Главная страницаРис. 2 Отображение таблицы Чеков

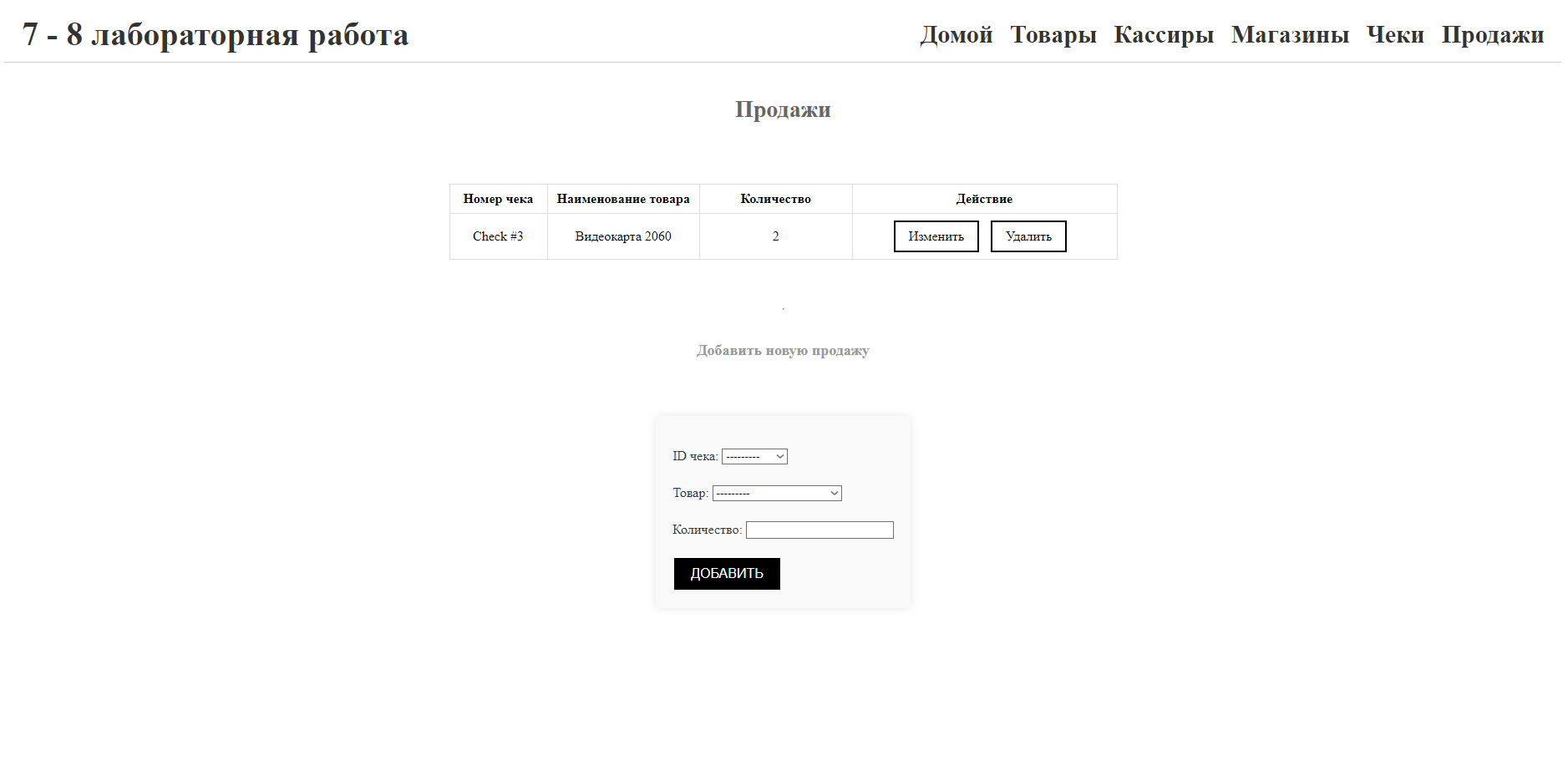


Рис. 3 Отображение таблицы Продаж

Вывод

В данной лабораторной работе были приобретены навыки работы с фреймворком Django. В ходе работы была сделана база данных с 5 таблицами, связанных между собой, содержащие информацию о чеке, продаже, продавце, магазине и товаре.

Код программы

models.py

from django.db import models

from django.contrib import admin

import datetime

from django.utils import timezone

from django.db import models

from django.utils import timezone

class Item(models.Model):

name = models.CharField(max\_length=255)

def \_\_str\_\_(self):

return self.name

class Cashier(models.Model):

name = models.CharField(max\_length=100)

phone\_number = models.CharField(max\_length=20)

def \_\_str\_\_(self):

return self.name

class Store(models.Model):

name = models.CharField(max\_length=255)

address = models.CharField(max\_length=255)

def \_\_str\_\_(self):

return self.name

class Check(models.Model):

cashier = models.ForeignKey(Cashier, on\_delete=models.CASCADE)

date\_time = models.DateTimeField()

total\_sum = models.DecimalField(max\_digits=999999, decimal\_places=2)

store = models.ForeignKey(Store, on\_delete=models.CASCADE)

def \_\_str\_\_(self):

return f'Check #{self.pk}'

class Sale(models.Model):

check\_id = models.ForeignKey(Check, on\_delete=models.CASCADE)

item = models.ForeignKey(Item, on\_delete=models.CASCADE)

quantity = models.PositiveIntegerField()

def \_\_str\_\_(self):

return f'Sale #{self.pk}'